HARDWARE-BASED COUNTERMEASURES AGAINST PHYSICAL ATTACKS

DISSERTATION

---

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

---

by Tobias Schneider
Bochum, October 2016

Copyright © 2016 by Tobias Schneider. All rights reserved.
Printed in Germany.

Tobias Schneider
Place of birth: Oberhausen, Germany
Author’s contact information:
`tobias.schneider-a7a@rub.de`
http://www.emsec.rub.de/chair/_staff/Tobias_Schneider/

Thesis Advisors:	**Prof. Dr.-Ing. Tim Güneysu** DFKI and University of Bremen, Germany **Priv.-Doz. Dr. Amir Moradi** Ruhr-Universität Bochum, Germany
External Referee:	**Pr. François-Xavier Standaert** Université catholique de Louvain, Belgium
Thesis submitted:	October 25, 2016
Thesis defense:	February 24, 2017
Last revision:	January 24, 2018

Abstract

In recent years, an increasing number of newly-designed products are equipped with small processing devices. Even though these *embedded systems* are computationally-limited, they are often required to perform complex cryptographic operations to achieve certain security goals, e.g., confidentiality or authenticity. Thus, for these critical and heavy computations, hardware-based accelerators (i.e., Application Specific Integrated Circuit (ASIC) or Field Programmable Gate Array (FPGA)) are utilized to lift the burden from the microprocessor and enable a correct and secure functionality in a constrained environment. Due to their physical accessibility, these cryptographic implementations need to be protected against physical attackers which can *passively* measure the physical characteristics of the device and *actively* inject faults in the computation. However, the secure and efficient integration of countermeasures against physical attacks is a non-trivial process, especially for the aforementioned hardware-based accelerators.

The first focus of this thesis is on evaluation methodologies for Side-Channel Analysis (SCA). This *passive* type of attack has been used in the past to break security-critical embedded systems and poses a severe threat to any unprotected cryptographic implementation. Therefore, the development of countermeasures is of uttermost practical relevance. Evaluation methodologies are an essential part of this design process as they represent a valuable tool to assess the vulnerability of a prototype. The results of these evaluations can indicate implementation errors and are used to compare the efficiency of different types of countermeasures. Therefore, they help to significantly improve the final product. There are various different evaluation techniques which can be roughly categorized into *attack-based*, *test-based*, and *information-theoretic* evaluations. This thesis includes contributions to all three categories. In particular, the theory behind the widespread leakage assessment methodology based on Welch's t-test is extended to allow correct and efficient leakage assessment at higher orders. Furthermore, the computation of correlation-based attacks, which account for a majority of attack-based evaluations, is significantly improved with the introduction of robust and one-pass algorithms to compute Pearson's correlation coefficient at arbitrary orders. As a third aspect, advanced statistical tools are applied to extend the evaluation capabilities of the information-theoretic metric to enable a more thorough leakage quantification of masked hardware designs. In addition, these tools can be also utilized to improve attack-based evaluation techniques which require density estimation.

The second focus of the thesis is on the design of novel hardware-based countermeasures against physical attacks. Specific physical phenomena, e.g., glitches, make the secure integration of countermeasure, especially masking schemes, on hardware-based devices a challenging process. The concept of Threshold Implementation (TI) represents one solution which enables the design of efficient masked hardware circuits. However, there are still limitations and open problems related to TI. This thesis includes contributions to three unsolved problems of hardware-based countermeasures against physical attacks. For one, the secure conversion between Boolean- and arithmetic-masked values, which has been thoroughly evaluated for software implementations, is examined. In this thesis, the first protected hardware design, which has been practically verified using the aforementioned leakage assessment methodology, is proposed. Furthermore, the thesis includes a study of cryptographic 8-bit Sboxes which are easy to protect against side-channel analysis in hardware. To this end, six different 8-bit Sboxes are presented and form a basis for future research on high-security block ciphers with intrinsic protection against physical attacks. As a last aspect, a combined countermeasure against both passive and active attacks is considered. This is a very important problem as secure systems need to incorporate hybrid countermeasures against both types of physical attacks. However, the focus of a majority of previous publications lies on only one of these two types of attack, while excluding the complementary threat. Therefore, the presented construction can be seen as a foundation which can drive further research in this area.

Keywords.

Physical Attacks, Side-Channel Analysis, Masking, Threshold Implementation, Fault Injection Attacks, Error-Detecting Codes, t-test, Mutual Information, Pearson's Correlation Coefficient

Kurzfassung

Die Zahl von Endgeräten, die mit kleinen Prozessoren ausgestattet sind, ist in den letzten Jahren stark gestiegen. Obwohl diese *eingebetteten Systeme* nur über eine sehr beschränkte Rechenkraft verfügen, müssen sie oft komplexe, kryptographische Operationen durchführen, um die benötigten Sicherheitsanforderungen zu erreichen. Daher werden für diese kritischen und schwierigen Berechnungen hardwarebasierte Beschleuniger (ASIC oder FPGA) eingesetzt, die eine korrekte und sichere Funktionalität in einer beschränkten Umgebung sicherstellen müssen. Da diese kryptographischen Implementierungen für Angreifer oft einfach zugänglich sind, müssen sie gegen physikalische Angreifer, die zum einen *passiv* die physikalischen Eigenschaften des Gerätes messen und zum anderen *aktiv* einen Fehler in der Berechnung herbeiführen können, geschützt werden. Allerdings ist die sichere und effiziente Integration von Schutzmaßnahmen gegen physikalische Angriffe nicht trivial. Dies gilt im Besonderen für die oben genannten hardwarebasierten Beschleuniger.

Der erste Schwerpunkt dieser Dissertation liegt auf Evaluationsmethoden für Seitenkanalanalysen. Dieser *passive*, physikalische Angriffstyp wurde in der Vergangenheit benutzt um sicherheitskritische eingebettete Systeme zu brechen und stellt für jede ungeschützte, kryptographische Implementierung eine ernsthafte Bedrohung dar. Folglich ist die Entwicklung von Schutzmaßnahmen von hoher praktischer Relevanz. Evaluationsmethoden sind ein essentieller Teil dieses Designprozesses, da sie ein wertvolles Werkzeug sind, um die Verwundbarkeit eines Prototypen zu bestimmen. Die Ergebnisse einer Evaluation offenbaren Implementierungsfehler und können benutzt werden, um verschiedene Schutzmaßnahmentypen zu vergleichen. Daher helfen sie die Qualität des finalen Produktes erheblich zu steigern. Es gibt verschiedene Evaluationstechniken, welche grob in die drei Kategorien *angriffsbasierte*, *testbasierte* und *informationstheoretische* Methoden eingeteilt werden können. Diese Dissertation beschreibt Beiträge zu allen drei Kategorien. Zum einen werden die theoretischen Grundlagen hinter der weitverbreiteten Evaluationsmethode, die auf dem Welch-Test basiert, erweitert, um eine korrekte und effiziente Bewertung auf höheren Ordnungen zu ermöglichen. Des Weiteren wird die Berechnung von korrelationsbasierten Angriffen, die einen Hauptteil der angriffsbasierten Evaluationen ausmachen, signifikant verbessert durch die Einführung von robusten und effizienten Berechnungsalgorithmen, die es ermöglichen die Pearson-Korrelation für beliebige Ordnungen zu berechnen. Als dritter Aspekt werden ausgefeilte, statistische Werkzeuge benutzt, um die Evaluationsmöglichkeiten der informationstheoretischen Metrik zu erweitern und so eine eingehendere Quantifizierung von maskierten Hardwaredesigns zu ermöglichen. Zusätzlich können diese Werkzeuge auch für bestimmte angriffsbasierte Methoden verwendet werden.

Der zweite Schwerpunkt der Dissertation liegt auf dem Design von neuartigen, hardwarebasierten Schutzmaßnahmen gegen physikalische Angriffe. Bestimmte physikalische Phänomene erschweren die sichere Einbettung von Schutzmaßnahmen, insbesondere Maskierungsverfahren, in hardwarebasierten Endgeräten. Das Prinzip der Threshold Implementierung (TI) stellt eine Möglichkeit dar, effiziente und maskierte Schaltkreise zu erstellen. Jedoch hat dieser Ansatz bestimmte Beschränkungen und Probleme. Diese Dissertation beschreibt Beiträge zu drei offenen Problemen von hardwarebasierten Schutzmaßnahmen gegen physikalische Angriffe. Zum einen wird die sichere Konvertierung zwischen boolesch-maskierten und arithmetisch-maskierten Werten, die für Softwareimplementierungen bereits ausführlich untersucht ist, betrachtet. In dieser Dissertation wird das erste geschützte Hardwaredesign vorgeschlagen, dessen Sicherheit praktisch mit den oben genannten Werkzeugen evaluiert wurde. Des Weiteren enthält die Dissertation eine Studie über kryptographische Sboxen, die einfach gegen physikalische Angriffe geschützt werden können. Zu diesem Zweck werden sechs verschiedenen Sboxen präsentiert, die eine Grundlage für weiterführende Forschung im Bereich von sicheren Blockchiffren mit intrinsischem Schutz gegen physikalische Angriffe bilden. Als letzter Aspekt wird eine kombinierte Schutzmaßnahme gegen passive und aktive Angriffe betrachtet. Dies ist eine relevante Problemstellung, da sichere Systeme hybride Schutzmaßnahmen gegen beide Angriffstypen voraussetzen. Jedoch liegt der Fokus eines Großteils von vorhergehenden Publikationen nur auf einem der beiden Angriffstypen. Daher kann die präsentierte Konstruktion als Grundlage gesehen werden, die weiterführende Forschung auf diesem wichtigen Gebiet ermöglichen wird.

Schlagworte.

Physikalische Angriffe, Seitenkanalanalyse, Maskierung, Threshold Implementierung, Fehlerinjektionsangriff, Fehlerkorrekturverfahren, t-Test, Transinformation, Pearson-Korrelation

Acknowledgements

This thesis is the result of three years of research at the Hardware Security Group (SHA) and the chair for Embedded Security (EMSEC) at Ruhr-Universität Bochum (RUB). It would not have been possible without the support of many remarkable individuals, some of which I would like to explicitly thank at this point. Firstly, I am grateful to Tim Güneysu for initially pushing me to pursue a PhD and putting trust in my capabilities. Thanks for being a terrific advisor by always allowing me great freedom in my projects and patiently listing to my concerns. Secondly, I would like to thank my other advisor Amir Moradi for taking me under his wing and teaching me the secrets of physical security. You have been a fantastic mentor and are one of the main reasons for the quality of this thesis. Thirdly, I would like to thank my external referee François-Xavier Standaert for taking time to read my thesis and attend my defense. Furthermore, I want to mention all members of (Ex-)SHA: Ingo von Maurich for advising my bachelor thesis and bringing me to SHA, Oliver Mischke for teaching me VHDL, Tobias Oder and Thomas Pöppelmann for opening my eyes to the wonderful field of lattice cryptography, Pascal Sasdrich for sharing an office with me and enduring the peculiarities of a fast-track PhD program together with me, and Alexander Wild for many valuable discussions no matter how nonsensical the topic. This thank extends to all members of EMSEC of which there are too many to list them all: Falk Schellenberg for answering any questions I had related to fault attacks and introducing me to the world of university sports, Bastian Richter for his extensive engineering knowledge, Georg Becker for always taking time out of his day to keep the PhD students motivated, Pawel Swierczynski for guiding me during my first months in the group, and the whole EMSEC football team. You created a great working environment in which it was easy to learn and thrive. Additionally, gratitude is owed to EMSEC's technician Horst Edelmann and team assistant Irmgard Kühn since without them I would have often been lost in the jungle of German bureaucracy. I also would like to thank Gregor Leander for temporarily welcoming me in his group for symmetric cryptography (SymCrypt), Christof Beierle and Thorsten Kranz for sharing their office with me and for the daily game of darts. A further thank you is owed to Sebastian Faust and Clara Paglialonga for a (seemingly never-ending) collaboration which condemned us to countless hours in front of a whiteboard. Furthermore, I would like to thank all members of the UbiCrypt (Ubiquitous Cryptography) Research Training Group (DFG GRK 1817/1) for the often productive seminars and the refreshing annual research retreats. Especially, I am thankful to the Ubicrypt coordinator Dominik Baumgarten for answering my many questions. Last but not least, I want to thank my family and friends for keeping me sane and making my life easy.

Table of Contents

Imprint . . . iii
Abstract . . . v
Kurzfassung . . . vii
Acknowledgements . . . ix

I Preliminaries 1

1 Introduction 3
1.1 Motivation . . . 3
1.2 Summary of Research Contribution . . . 5
  1.2.1 Evaluation Methodologies . . . 5
  1.2.2 Hardware-Based Countermeasures . . . 7
1.3 Structure of this Thesis . . . 9

2 Side-Channel Attacks and Countermeasures 11
2.1 Countermeasures . . . 11
  2.1.1 Masking . . . 12
  2.1.2 Hiding . . . 13
2.2 Attacks . . . 13
  2.2.1 Higher-Order Attacks . . . 14

II Evaluation Methodologies for Side-Channel Countermeasures 17

3 Background: Evaluation Methodologies 19
3.1 Statistical Background . . . 19
  3.1.1 Statistical Moments . . . 19
  3.1.2 Normal Distribution . . . 21
3.2 Iterative Computation . . . 21

4 Leakage Assessment Methodology 25
4.1 Introduction . . . 25
  4.1.1 Contribution . . . 26
4.2 Background . . . 27
4.3 Methodology . . . 28
  4.3.1 Order of the Test . . . 30

4.4 Efficient Computation . . . 31
4.4.1 Incremental One-Pass Computation of All Moments . . . 32
4.4.2 Variance of Preprocessed Traces . . . 32
4.4.3 Parallel Computation . . . 34
4.5 Multivariate Evaluation . . . 34
4.6 Case Studies . . . 37
4.6.1 Framework . . . 38
4.6.2 Case Study: Microcontroller . . . 40
4.6.3 Case Study: FPGA . . . 41
4.7 Conclusion . . . 43

5 Robust and One-Pass Parallel Computation of Correlation-based Attacks 45
5.1 Introduction . . . 45
5.1.1 Contribution . . . 46
5.2 Notations . . . 46
5.3 Univariate CPA . . . 47
5.3.1 Univariate Higher-Order CPA . . . 48
5.3.2 Numerator . . . 49
5.3.3 Denominator . . . 51
5.4 Multivariate CPA . . . 51
5.4.1 Numerator . . . 52
5.4.2 Denominator . . . 53
5.5 Moments-Correlating DPA . . . 54
5.5.1 Numerator . . . 55
5.5.2 Denominator . . . 56
5.5.3 Reuse of Sums . . . 57
5.6 Evaluation . . . 59
5.7 Conclusion . . . 61

6 Advanced Tools for Side-Channel Leakage Estimation 63
6.1 Introduction . . . 64
6.1.1 Contribution . . . 64
6.2 Background . . . 66
6.2.1 Histograms . . . 66
6.2.2 Kernels . . . 67
6.2.3 Gaussian Density Estimation . . . 67
6.3 New Proposals . . . 68
6.3.1 Exponentially Modified Gaussian . . . 68
6.3.2 Pearson Distribution System . . . 69
6.3.3 Shifted Generalized Lognormal . . . 70
6.3.4 Coverage of Pearson and SGL . . . 72
6.3.5 Computational Complexity . . . 72
6.4 Simulated Experiments . . . 72
6.5 Practical Case Studies . . . 74
6.5.1 Profiled Evaluations and Attacks . . . 76

6.5.2 Non-Profiled Attacks . . . 81
6.5.3 Selection of Tools . . . 82
6.6 Conclusion and Future Work . . . 84

III Advanced Countermeasures against Physical Attacks 85

7 Background: Countermeasures 87
7.1 Threshold Implementations . . . 87

8 Arithmetic Addition over Boolean Masking in Hardware 91
8.1 Introduction . . . 91
8.1.1 Related Work . . . 92
8.1.2 Contribution . . . 92
8.2 Background . . . 93
8.2.1 Notation . . . 93
8.2.2 Ripple-Carry Adder . . . 93
8.2.3 Kogge-Stone Adder . . . 94
8.3 Implementation . . . 94
8.3.1 Ripple-Carry Adder (First-Order SCA-Resistant) . . . 95
8.3.2 Ripple-Carry Adder (Second-Order SCA-Resistant) . . . 96
8.3.3 Kogge-Stone Adder (First-Order SCA-Resistant) . . . 97
8.3.4 Kogge-Stone Adder (Second-Order SCA-Resistant) . . . 98
8.3.5 Comparison . . . 98
8.4 Analysis . . . 99
8.4.1 Ripple-Carry Adder . . . 99
8.4.2 Kogge-Stone Adder . . . 102
8.4.3 Higher-Order Security . . . 102
8.5 Conclusion and Future Work . . . 104

9 Strong 8-bit Sboxes with Efficient Masking in Hardware 105
9.1 Introduction . . . 105
9.1.1 Contribution . . . 106
9.2 Background . . . 107
9.2.1 Cryptanalytic Properties of Sboxes . . . 107
9.2.2 Construction of 8-Bit Sboxes . . . 109
9.2.3 TI of 4-bit Permutations. . . . 109
9.2.4 Design Architectures . . . 110
9.3 Threshold Implementation of Known 8-Bit Sboxes . . . 111
9.3.1 CLEFIA . . . 111
9.3.2 Crypton V0.5 . . . 112
9.3.3 Crypton V1 . . . 113
9.3.4 ICEBERG . . . 113
9.3.5 Fantomas . . . 113
9.3.6 Khazad . . . 114

9.3.7 Robin . . . 114
9.3.8 Scream V3 . . . 115
9.3.9 Whirlpool . . . 115
9.3.10 Implementation . . . 115
9.4 Finding TI-Compliant 8-Bit Sboxes . . . 116
9.4.1 Feistel-Construction . . . 118
9.4.2 SPN-Construction with Bit-Permutations as the Linear Layer . . . 119
9.4.3 SPN-Construction with $\mathbb{F}_{16}$-linear Layers . . . 120
9.5 Results . . . 121
9.5.1 Selected Sboxes . . . 122
9.5.2 Comparison . . . 123
9.6 Conclusion and Future Work . . . 124

10 ParTI: Towards Combined Hardware Countermeasures 127
10.1 Introduction . . . 127
10.1.1 Related Work . . . 128
10.1.2 Contribution . . . 128
10.2 Background . . . 129
10.2.1 Error Detecting Codes . . . 129
10.2.2 Concurrent Error Detection . . . 130
10.3 Methodology . . . 130
10.3.1 Design Considerations . . . 131
10.3.2 Attacker Model . . . 131
10.3.3 Code Selection . . . 132
10.3.4 Threshold Implementations with Error Detecting Codes . . . 133
10.3.5 Security Analysis . . . 138
10.4 Case Study: LED . . . 142
10.4.1 Cipher Description . . . 143
10.4.2 Design and Implementation . . . 144
10.4.3 Area Comparison . . . 149
10.4.4 Resistance against SCA . . . 150
10.4.5 Resistance against FI . . . 150
10.5 Conclusions and Future Work . . . 152

IV Conclusion 153

11 Conclusion and Future Work 155
11.1 Conclusion . . . 155
11.2 Future Work . . . 157
11.2.1 Physical Adversary Model . . . 157
11.2.2 Efficient and Secure Randomness Generation . . . 157
11.2.3 Masking in Hardware with Less Shares . . . 158
11.2.4 Physically Secure Cipher . . . 158

V Appendix 159

12 Specific Formulas 161
12.1 Univariate Two-Pair Iterative 161
12.1.1 Central Sums 161
12.1.2 Adjusted Central Sums 162
12.2 Univariate Incremental 163
12.2.1 Mean 163
12.2.2 Central Sums 163
12.2.3 Adjusted Central Sums 164
12.3 Central Moments from the Raw Moments 165
12.4 Mean and Variance of the Preprocessed Measurements 165
12.5 Univariate Correlation 166
12.6 Bivariate Second-Order Evaluation 166
12.6.1 Two-Pair Iterative 166
12.6.2 Incremental 168
12.6.3 Correlation from the Raw Moments 169
12.6.4 Third Order 169
12.6.5 Fourth Order 170
12.6.6 Fifth Order 170

13 Second-Order Threshold Implementation of RCA and KSA 171
13.1 Second-Order RCA 171
13.1.1 Carry (1. Step) 171
13.1.2 Carry (2. Step) 171
13.2 Second-Order KSA 172
13.2.1 AND (1. Step) 172
13.2.2 AND/XOR (1. Step) 172

14 Specifications of the Selected Sboxes 173
14.1 Algebraic Degree 173
14.2 Look-up tables 174

15 Shared Functions of the Sbox 177
15.1 Shared Functions Sbox 177

Bibliography 181

List of Abbreviations 205

List of Figures 207

List of Tables 209

About the Author 211

Publications and Academic Activities **212**

Part I

Preliminaries

Chapter 1

Introduction

This chapter contains a brief introduction to the field of physical security. We establish the threat of physical attacks for hardware-based cryptographic implementations and thus motivate the necessity of thorough leakage evaluation methodologies and efficient countermeasures. These two aspects of physical security form the basis of this thesis, and we give a summary of our research contribution related them. In addition, the structure of this thesis is described.

Contents of this Chapter

1.1 Motivation **3**
1.2 Summary of Research Contribution **5**
1.3 Structure of this Thesis **9**

1.1 Motivation

Embedded systems are an essential part of contemporary industrial and commercial products. Recent trends like the Internet of Things (IoT) [AIM10] have created a seemingly never-ceasing demand for small and pervasive computing platforms which are embedded into commodity hardware. This process increases the connectivity of newly-designed devices and enables the development of revolutionary new functionality, e.g., vehicular communications systems [Pap16]. However, certain products, which operate in security-critical applications, process sensible data and, therefore, need to incorporate dedicated measures to protect the users. To this end, cryptography is often utilized and included in embedded systems to achieve the required security properties, e.g., privacy, confidentiality, or authenticity. Depending on the selected cryptographic primitive, the required cryptographic computations can result in a non-negligible performance overhead due to the limited computational power of a majority of embedded processors. Usually, these processors are selected to be very small and cost-efficient and thus are not suited to efficiently perform complex operations. In addition, certain use cases place further constrains on the cryptographic implementations. These include a specific level of throughput to process large amounts of data in a small time frame or an extremely low latency for applications in the Tactile Internet [SAD$^+$16] which tries to achieve more natural human-to-machine interaction. For these scenarios, dedicated hardware modules are needed to take the burden off the microprocessor and significantly speed up the cryptographic operations.

ASICs are considerably more efficient and powerful than equivalent software implementations on an embedded Microcontroller (μC). They are heavily optimized to perform one specific task and therefore can provide superior throughput, energy efficiency, and low latency which are all important metrics for embedded systems. In addition, ASICs are also more cost-efficient assuming a sufficient quantity, but their initial development costs are significantly higher than a standard software implementation on an off-the-shelf microprocessor. Therefore, for small number of units integrating an ASIC into an embedded system is too costly. For these scenarios, FPGAs are a sound alternative. They are commonly based on reprogrammable lookup tables to emulate dedicated hardware circuits and have become increasingly popular in recent years due to their flexibility and lower development costs. Similar to software implementations, they can be bought off-the-shelf with the additional benefit over ASICs that it is possible to reprogram them in the field. All three platforms (i.e., ASIC, FPGA, and μC) have their field of application for embedded systems. However, as described before critical operations (e.g., cryptography) are increasingly moved from software to hardware-based platforms to ensure a reliable and high performance.

Most of the modern cryptographic algorithms have been thoroughly evaluated to verify their theoretical security, making traditional cryptanalysis not an imminent threat to embedded systems. However, the specific implementation of a secure cryptographic algorithm incorporated in these systems is still vulnerable to physical attacks due to their physical accessibility. A physical attacker, which can *passively* measure the physical characteristics of the device during computation, can apply SCA to extract sensitive information from side-channels, e.g., timing [Koc96]. Complementary, an *active* attacker, which is able to inject a fault into the system, can tamper with the computation and use Fault Injection (FI) attacks to derive the secret key from the faulty output of system [BS97]. These two kinds of attack do not try to break the secure algorithm but instead target the vulnerabilities of the implementation. Therefore, they are very powerful in practice and have been used in the past to break numerous widely-used embedded systems, e.g., locking systems [SDK$^+$13, EKM$^+$08]. Without dedicated countermeasures, an unprotected implementation of a standardized cryptographic algorithm (e.g., AES) is highly likely to be vulnerable to physical attacks.

Due to the severe threat of physical attacks, it is of uttermost importance for embedded systems to include dedicated countermeasures to thwart them. To this end, various different types of countermeasures against SCA and FI have been proposed. A majority of SCA protection schemes focuses either on reducing the Signal-to-Noise Ratio (SNR) (i.e., *hiding* schemes [VMKS12]) or randomizing the internal processed values (i.e., *masking* schemes [CJRR99]). FIs are mostly thwarted by introducing a form of redundancy (spatial or temporal) into the computation to detect faults and prevent the release of the faulty output. These countermeasures aim to make an attack very hard to conduct in practice by increasing the number required measurements and requiring more sophisticated types of fault. An embedded system needs to incorporate countermeasure against both passive and active attacks. Surprisingly, most academic publications regarding countermeasures focus on protecting a system against only one of the two attack vectors while excluding the other. Admittedly, this approach simplifies the process of designing and evaluating a new scheme, but it also neglects possible negative interactions between combinations of countermeasures. Since the integration of any countermeasure comes with a non-negligible performance overhead, the selected countermeasures needs to be finely tuned to the target algorithm and platform. The integration requires particular care to ensure that the countermeasures are implemented correctly and provide the required level of security, which can be non-trivial task in some scenarios. Especially the implementation of masking schemes in dedicated hardware circuits poses a challenge due to certain physical phenomena, e.g., glitches [MPO05].

The correctness and efficiency of a protected implementation can be assessed with various evaluation methodologies. Especially, the security evaluation against passive attackers has sparked different approaches. They can be categorized in *attack-based*, *test-based*, and *information-theoretic* evaluation methodologies and each category has different ranges of application. Depending on the use case, the evaluation process can result in non-negligible time requirements to ensure that every vulnerability is covered by the process. Nevertheless, it is a very important part of the design cycle of protected implementations. Thorough evaluations can help the designer find implementation bugs early in the design phase and increase the confidence in the security of the final product. The importance of practical evaluations is highlighted by examples of published countermeasures which are later found to be vulnerable [SM15a]. Furthermore, it allows to compare the efficiency and security of different countermeasure, which enables the designer to pick the best suited countermeasures for the final product. Evaluation techniques are also applied by evaluation labs to analyze a product and award security certificates where appropriate.

1.2 Summary of Research Contribution

The research contributions described in this thesis are all related to protecting hardware-based designs for FPGAs and ASICs against physical attacks. A majority of the presented results are part of the conference and journal publications [SMG15b, SM15a, SMG16c, SM16, SMG16b, BGG$^+$16a, SMSG16b]. Each publication covers a specific aspect of the main topic and forms the basis for a chapter of this thesis. These chapters can be categorized into two thematic groups. The first group describes advances in leakage evaluation methodologies, which can be used to assess the vulnerability of an implementation against SCA. A special focus of this group lies on the analysis of masked hardware circuits. The second group contains chapters related to novel hardware-based countermeasure against physical attacks. We apply our knowledge of leakage evaluation methodologies to create protected hardware designs secure against passive and active physical attackers. In the following, we briefly summarize the important problems which are addressed in each category and highlight our contribution.

1.2.1 Evaluation Methodologies

As described in the previous section, security evaluation is an important part of the development process for many security-critical embedded systems. However, depending on the selected evaluation technique and target device, this evaluation process can be quite cumbersome. In the early years of side-channel research, it was customary to rely on attacks for evaluation, i.e., testing a device by conducting a range of different attacks. This approach suffers from an ever rising number of attacks and increasingly complex countermeasures, which makes comprehensive evaluations a difficult and time-consuming task. Alternatively, various test methodologies have been proposed. One of the most popular techniques of the recent years has been proposed by Goodwill *et al.* in [GJJR11] and is based on the renowned t-test. Especially, the non-specific test offers a very efficient and generic method to assess the leakage of a target device. However, it can lack informativeness and suffers from the occurrence of false positives if the measurement framework is deficient. A different evaluation methodology based on an information-theoretic metric was introduced by Standaert *et al.* in [SMY09] and later refined in [RSV$^+$11]. Nevertheless, every of these three evaluation categories (i.e., attack-based, test-based, information-theoretic) has its own advantages and range of applications. In this thesis, we present one contribution to each of the three types with the goal to overall improve the current state-of-the-art of the evaluation process.

Leakage Assessment Methodology [SM15a, SM16]

Evoked by the increasing need to integrate side-channel countermeasures into security enabled commercial devices, evaluation labs are seeking a standard approach that enables a fast, reliable and robust evaluation of the side-channel vulnerability of the given products. To this end, standardization bodies such as NIST intend to establish a leakage assessment methodology fulfilling these demands. One of such proposals is the Welch's t-test, which is being put forward by Cryptography Research Inc., and is able to relax the dependency between the evaluations and the device's underlying architecture. It has been used in a couple of research publications [BGG$^+$14, LMW14, BGN$^+$14b, SMG15b, SMMG15b, MH15, SMMG15a, WMG15] to investigate the efficiency of the proposed countermeasures, but without extensively expressing the challenges of the test procedure. In this context, our contribution is manifold and addresses multiple unresolved problems related to this leakage assessment methodology. Firstly, we reformulate the underlying statistical concepts into a (hopefully) more understandable terminology to make the access to the test easier. Secondly, we extend the original instruction, which heavily focuses on conducting the test at the first order, to more complex use cases, i.e., multi- and univariate higher-order evaluations. Thirdly, we discuss the challenges of processing several million traces and propose robust one-pass algorithms which can be efficiently computed in parallel. Lastly, a measurement framework for software and hardware evaluations is described which allows a correct and efficient collection of measurements. It is utilized in two practical cases studies of supposedly secure designs and the resulting detected leakages serve to underline the practical relevance of both the testing methodology and our contributions.

Robust and One-Pass Parallel Computation of Correlation-based Attacks [SMG16c]

Correlation-based attacks are an important part of physical security evaluations ever since the initial proposal of the basic Correlation Power Analysis (CPA) in [BCO04]. Over the years, many extensions to the original concept have been proposed including the usage of different leakage models and collisions. Moments-Correlating DPA (MC-DPA) [MS14] as the successor of the Correlation-Enhanced Power Analysis Collision Attack [MME10] is the latest addition to the family of correlation-based attacks and relaxes the necessity of a hypothetical leakage model. However, the large number of different types of attack and leakage models make a thorough evaluation with this approach very time-consuming. This is amplified by the increasing complexity of new countermeasures which require complex evaluations at multiple orders and points in time. To this end, we propose new algorithms to compute Pearson's correlation coefficient and thus any correlation-based side-channel evaluation. These algorithms utilize the concept of incremental computation and thus process every measurement only once even for higher-order evaluations. This helps to significantly accelerate the evaluation process which can be easily split over multiple processing units due to the parallelization capabilities of our algorithms. In addition, the number of numerical problems is significantly reduced which has been a problem for simpler incremental approaches.

Advanced Tools for Side-Channel Leakage Estimation [SMSG16b]

An alternative evaluation methodology based on Mutual Information (MI) was introduced in [SMY09] and further refined in [RSV$^+$11]. The information-theoretic metric aims to enable a fair assessment of the security level of protected implementations and to allow designers to discuss security and performance implications for their implementations on a sound basis. However, the computation of the metric requires the estimation of leakage distributions (similar to certain attacks, e.g., Template Attack (TA) [CRR02] and Mutual Information Analysis (MIA) [GBTP08]) and is limited to specific mixtures of orders. Therefore, we give three contributions to improve this evaluation methodology. Firstly, we extend the evaluation toolbox with three new Probability Density Function (PDF) estimation tools and discuss their benefits and limitations over the commonly-used alternatives (i.e., Gaussian templates and kernel-based density estimation). Secondly, these tools are applied to compute the information leakage of a masked hardware design. In particular, we demonstrate how to use the tools to extend the capabilities of the metric to include new mixtures of orders. Lastly, we investigate how the tools can be used for profiled and non-profiled attack-based evaluations by conduction a TA and MIA on the masked hardware design.

1.2.2 Hardware-Based Countermeasures

Given the severe threat that physical attacks pose to cryptographic implementations, extensive research has been expended to create measures to thwart this type of attack. However, there are still unresolved problems, especially concerning secure hardware designs, which demand further investigations to make protected implementations more secure and more efficient. In this thesis, we propose solutions to three major issues regarding hardware-based countermeasures which have been mostly neglected so far. This includes the conversion between Boolean- and arithmetic-masked values, the efficient masked implementation of 8-bit Sboxes in hardware, and a concept for combined countermeasures against both passive and active physical attacks.

Arithmetic Addition over Boolean Masking in Hardware [SMG15b]

The conversion between arithmetic- and Boolean-masked values is a common problem for masked algorithms which require both arithmetic and Boolean operations. For instance, ARX-based designs consist of the three operations integer addition, bit rotation and XOR and are the foundation for many cryptographic constructions, e.g., block ciphers [Miy90, FLS$^+$10]. In [Gou01], Goubin proposed two algorithms to convert between these two masking types and thus enabled the first secure implementations. Over time, many publications have extended these initial ideas to create more efficient conversion algorithms [CGTV15] for different fields of application. In addition, Karroumi *et al.* introduced the concept of *blinded addition* which denotes the secure arithmetic addition of Boolean-masked inputs [KRJ14]. Due to the absence of a second conversion step, this approach can be advantageous if the target algorithm does not contain consecutive arithmetic additions. However, all the aforementioned schemes were designed for software-like platforms and are not optimized for the implementation on a hardware-based device. This results in designs which are not secure due to hardware-exclusive phenomena (e.g., glitches) and do not take full advantage of the target's platform capabilities (e.g., inexpensive bit operations). To the best of our knowledge, there is only one hardware-based solution by Golic [Gol07]. However, it was published before the effect of glitches on the security of an implementation [MPO05] became widely-known and therefore suffers from the same security issues as the former software-oriented approaches. In this thesis, we propose to use the proven concept of TI to solve this problem and present two efficient designs for blinded addition in hardware. Both proposals are optimized for different fields of applications including a lightweight and a high-performance design. We practically verified their security using the aforementioned leakage assessment methodology using the t-test and compared them to the existing schemes. Our constructions outperform the existing algorithms in terms of fresh randomness and throughput with the added benefit of providing resistance against side-channel analysis.

Strong 8-bit Sboxes with Efficient Masking in Hardware [BGG$^+$16a]

Masking in hardware circuits is commonly realized using the concept of TI which can provide sufficient security even in the presence of glitches. However, creating an efficient TI for a standard block cipher is still challenging due to the high algebraic degree of commonly-used cryptographic Sboxes. For smaller Sboxes exhaustive search can be used to decompose the Sbox into multiple functions of a lower degree [BNN$^+$15] which helps to significantly improve the efficiency of the masked implementation. However, this approach is not viable for byte-oriented ciphers, e.g., AES [Pub01], due to size of the resulting search space. There are some implementations of the renowned Sbox of the AES, which all require additional randomness and do not achieve the same the efficiency as TI of smaller Sboxes, e.g., PRESENT [BKL$^+$07]. In this thesis, we present an alternative approach to solve the problem of inefficient TI for larger Sboxes. Instead of finding a TI representation for a good cryptographic Sbox, we reverse the development process and try to construct good cryptographic Sboxes from round functions which have an efficient TI realization. In particular, we focus on 8-bit Sboxes and use 4-bit functions as building blocks for the rounds. We compare our newly constructed Sboxes with other 8-bit Sboxes regarding their cryptographic properties and their implementation efficiency. In terms of cryptographic properties, our constructions are on-par with the selected 8-bit Sboxes excluding AES. However, their area consumption, throughput, and latency is significantly better than of the considered examples. These results are especially useful for designers of block ciphers as they promote the design of high-security block ciphers with intrinsic protection against physical attacks

ParTI: Towards Combined Hardware Countermeasures [SMG16b]

Without the integration of dedicated countermeasures against both active and passive physical attacks, embedded systems are likely vulnerable to a physical attacker and cannot be utilized to process sensitive information. Over the years, many different countermeasures against each type have been proposed. However, a majority of these publications focuses on only one of the two types of physical attacks and the proposed countermeasure is not evaluated in the complementary context. This results in inefficient combinations of independent countermeasures and in the worst-case can negatively affect the security due to one countermeasure canceling another. First approaches were made to design a combined countermeasure based on coding theory by Bringer *et al.* in [BCC^{+}14]. However, their solutions are optimized for software implementations and cannot be easily transferred to a dedicated hardware circuits without sacrificing security or efficiency. In this thesis, we present a new methodology to design hardware-based implementations protected against both types of physical attacks. To this end, the concept of TI is combined with the capability of error-detecting codes. We apply our methodology in a case study targeting the LED cipher [GPPR11] and practically verify the side-channel security of our design. The final design trades a small increase in area for significantly better protection against realistic fault models compared to more simplistic approaches. Our work can be seen as a sound basis for further research in the previously neglected field of hardware-based combined countermeasures.

1.3 Structure of this Thesis

The thesis is split into four major parts. Initially, a short introduction and general background information regarding physical attacks is given in the first part *Preliminaries*. This is followed by two result parts in which the main contributions of this thesis are presented. The last part contains the conclusion and discusses future work. In the following, the detailed structure of each part is given.

Preliminaries In this section, we give a brief overview of the current state of research in physical attacks and countermeasures and establish the contributions of this thesis. We revise common attacks and countermeasures for SCA.

(1) Introduction

(2) Physical Attacks and Countermeasures

Methods for Evaluation of Side-Channel Countermeasures In this part, we present our research contributions regarding different side-channel evaluation methodologies. The background chapter introduces incremental computation and basic statistical concepts regarding statistical moments and distributions. This is followed by our research contribution in which we describe our advances in the three categories of evaluation methodologies.

(1) Background

(2) Leakage Assessment Methodology

(3) Robust and One-Pass Parallel Computation of Correlation-based Attacks

(4) Advanced Tools for Side-Channel Leakage Estimation

Advanced Countermeasures Against Physical Attacks In Part III, we discuss our contributions to the design of countermeasures against physical attacks. The background chapter introduces the concept of TI in detail. Subsequently, we present our results concerning the three main aspects:

(1) Background

(2) Arithmetic Addition over Boolean Masking in Hardware

(3) Strong 8-bit Sboxes with Efficient Masking in Hardware

(4) ParTI: Towards Combined Hardware Countermeasures

Conclusion In this part, we summarize our research contributions and give directions for future research building on our results.

Chapter 2

Side-Channel Attacks and Countermeasures

Physical attacks are a severe threat to cryptographic implementations. Especially, side-channel analysis has been used in the past to successfully attack numerous security-critical embedded systems. In this chapter, we give a brief introduction about side-channel attacks and countermeasures.

Contents of this Chapter

2.1 Countermeasures **11**
2.2 Attacks **13**

Side-channel analysis describes a prominent type of passive physical attack. It is based on the analysis of the measurable physical characteristics of the target device during the computation. The most commonly-used characteristics include the timing behavior of the implementation [Koc96] and the power consumption during sensitive operations [KJJ99]. After obtaining enough measurements, a side-channel attacker uses various statistical methods to derive secret information from the measured traces. Ever since the first publication by Paul C. Kocher [Koc96], different types of side-channel attacks and countermeasures have been proposed. In the following, we briefly introduce the two most common countermeasure types (i.e., *masking* [CJRR99, NRR06, ISW03], and *hiding* [VMKS12, WMG15]) and shortly discuss the sophisticated attacks which are needed to attack implementation protected by these countermeasures.

2.1 Countermeasures

The design of efficient and secure countermeasures against side-channel analysis is an essential aspect of side-channel research. To this end, various different schemes have been proposed aimed to protect an implementation and make any side-channel attack impossible to conduct in practice. Usually, this is achieved by increasing the number of required measurements for a successful attack up to the point where it becomes practically unfeasible. Most schemes can be categorized into either *masking* or *hiding* schemes. In addition, *re-keying* schemes provide security by frequently updating the used secret key and reducing the security of multiple iterations to the security of only one [ABF13]. Some countermeasures possess a strong mutual interaction and are often combined to achieve better efficiency which is an important metric given that most countermeasures come with a non-negligible implementation overhead [MW15].

2.1.1 Masking

Masking schemes rely on randomizing the processed sensitive internal values to randomize the resulting side-channel leakage and thus thwarting attacks targeting these values. Due to its sound theoretical foundation, masking can provide provable security to a certain degree and has been thoroughly investigated [CJRR99]. There are different levels at which masking can be applied (e.g., algorithm-level [NRR06] or gate-level [LMW14]) and it needs to be adapted to the target platform. As shortly discussed in the introduction, this is an especially challenging task for masking in hardware-based implementation which is the focus of this thesis.

To randomize the processed values every sensitive variable x is split up into multiple shares which we denote as x^i for $1 \leq 1 \leq s$ where s denotes the total number of shares. In basic masking scheme these shares are generated using $s - 1$ random mask and the last share is computed to satisfy the following relation

$$x^1 \circ x^2 \circ \cdots \circ x^s = x \tag{2.1}$$

where $\circ$ denotes the group operation of the specific masking scheme. For example, Boolean (resp. arithmetic) masking is commonly applied to mask block ciphers and it relies on Exclusive OR (XOR) (resp. modular addition) as the group operation. The number of shares increases with the desired order of security depending on the utilized masking scheme.

The order of a masking scheme refers to its provided level of security. Given that the random masks used to generate the shares are independent and that the computations on these shares leak independently, only the joint distribution of all s shares leak information about x. For basic masking schemes the number of shares is directly derived from the order of the masking scheme d as $s = d + 1$. Therefore, to attack such a d-order masking scheme it is required to estimate the joint distribution of $d+1$ shares, i.e., perform a $d+1$-order attack. Against attacks of smaller order than $d + 1$ such a masking scheme does indeed provide provable security, i.e., these attacks are never successful independent of the number of measurements. However, this property only provides a practical benefit given a sufficient level of noise, since a $d + 1$-order attack can still be successful. In case the level of noise is indeed sufficient, the attack complexity increases exponentially with the order optimally preventing any $d + 1$-order attack in practice.

The type of masking scheme needs to be adapted to the operations of the target algorithm to achieve the best possible performance. Since every computation on the shares needs to be maintain the security assumption of the underlying masking scheme, i.e., $d + 1$-order leakage, it is important to select a group operation suited to the target operations. For Boolean functions, usually Boolean masking is chosen as it results in very efficient shared computations. Nevertheless, creating a masked algorithm in a secure and efficient way is a complex process.

There are different approaches which are used to prove the security of a masked algorithm. Recently, many publications rely on the probing model [ISW03] which assumes that the adversary can probe a finite number of wires inside the computation. If it is not possible to successfully attack the target using d probes, the system is d-th probing secure. This approach is widely accepted in the side-channel community and is suitable to prove the security of algorithms. However, side-channel analysis attacks the implementation of these algorithms and a provable-secure algorithm does not automatically imply a secure implementation. For software implementations it has been shown that distance-based leakages and other physical effects, which are not included in the proof, can severely reduce the security of an implementation [BGG+14]. For hardware-based designs glitches in the combinatorial logic were identified as a major source of secret-dependent leakage and are one of the main reasons why the secure realization of masking in hardware is challenging [MPG05, MPO05]. Therefore, the proofs of masked algorithms are only the initial steps to security and the security of the final implementations needs to be always practically evaluated.

2.1.2 Hiding

Contrary to masking schemes, hiding does not change the processed sensitive values and thus the secret-dependent leakage is still present in the measurements. Instead, hiding schemes aim to reduce the SNR of the measurements and in this way hampering the extraction of secret information from the traces. This is achieved by either reducing the signal with e.g., power equalization schemes [WMG15], or increasing the noise with e.g., shuffling [VMKS12]. Since the secret information is not masked, implementations relying on hiding schemes can still be successfully attacked with first-order attacks. However, the number of required measurements is strongly increased due to the decreased signal-to-noise ratio.

Nevertheless, the focus of this thesis is on masking schemes and we do not consider any type of hiding scheme in our protected designs. This is mainly due to the sound theoretical properties of masking schemes which enable security proofs up to a given order. Hiding schemes are still relevant to our results given that masking schemes require a sufficient level of noise to be effective. Therefore, combining our proposed masked designs with a hiding countermeasure would further increase the side-channel resistance of the implementation. Extending threshold implementations by including power equalization measures has been shown in [MW15] to provide a strongly increased resistance against side-channel analysis and their approach could be trivially copied for all of our proposed designs when targeting FPGAs.

2.2 Attacks

There are multiple criteria to distinguish between the ever-increasing number of different side-channel attacks. However, a majority of them follows the same underlying procedure of measuring a physical characteristic of the target device, using a leakage model to deduce the internal values for different key guesses, and applying a statistical distinguisher on the measurements and hypothetical values to derive the correct key [MOP07]. The most commonly-used side-channels consist of the timing behavior of the implementation, the power consumption, and the electromagnetic emanation of the device, due to their informativeness and ease of measurement. To model the hypothetical values usually either a bit model or the Hamming weight (resp. distance) of the values is used. Furthermore, a commonly attacks rely on Pearson's correlation coefficient to distinguish the correct key [BCO04, MME10, MS14]. Another way to distinguish side-channel attack is the use of a profiling device which enables the attacker to build templates prior to the attack phase [CRR02]. With this, the success probability can be significantly improved. Preprocessing the measurements can aid the success of a side-channel attack by for example decreasing the noise level and thus increasing the signal-to-noise-ratio. In some cases, i.e., higher-order attacks, preprocessing is even a strict requirement for the success of the attack.

2.2.1 Higher-Order Attacks

As described in the previous section, to successfully attack a masked implementation of order d a side-channel attacker needs to conduct an attack of order $d+1$ or higher. Therefore, the thorough evaluation of the side-channel resistance of a masked implementation requires that the evaluation process is also conducted at multiple orders to include the threat of higher-order attacks. In particular, the evaluation results for a d-order masked implementation should show that there is no leakage for orders smaller than $d+1$. Since this thesis focuses on masked designs and evaluation methodologies for masked designs, we provide a detailed foundation of higher-order attacks which are an important aspect of the later result parts.

For higher-order attacks, multiple points which correspond to the leakage of different shares x_i are combined using a specific combination function with the goal that their combination leaks information about the sensitive variable x. The optimum combination function is denoted as the centered product [PRB09, SVO$^+$10] and is defined as

$$Y = \prod_{i=1}^{d+1} (X_i - \mu_{X_i}) \tag{2.2}$$

for the variables $X_i, 1 \leq i \leq d+1$ to generate the combined variable Y. Since we exclusively use this specific combination function for our higher-order evaluations, we do not introduce other combination functions in this section, e.g., absolute difference. This combination of different points has to be computed for all measurements before the attack during a preprocessing step. With increasing order the preprocessing step can result in a non-negligible computation overhead and thwart the rapid evaluation of a new prototype. Therefore, some of our contributions focus on the robust and efficient computation of the centered product at arbitrary orders for different use cases to enable a fast development of masked implementations.

However, the computation complexity is not the main reason for the exalted difficulty of higher-order attacks. While it steadily increases with the order of the attack, the biggest limitation stems from the measurement complexity as it grow exponentially with the order of the attack for a sufficient level of noise. In [SVO$^+$10], the authors assume that the leakage samples can be modeled as

$$L_i = V_i + N_i \tag{2.3}$$

where V_i denotes the leakage of the processed value and N_i the normally distributed noise with mean 0 and variance σ_N^2. They show that the required number of measurements for a successful attack is proportional to $\left(\sigma_N^2\right)^{s/2}$ depending on the number of shares s and the noise variance σ_N^2. Meaning that if the measurement are sufficiently noisy, the attack complexity increases exponentially with the number of shares and correspondingly with the order of the attack. Since in practice the attacker has only access to a finite number of measurements, choosing the order of the masking scheme high enough can make the masked design practically secure against any realistically limited side-channel attacker. As previously mentioned, hiding schemes are often paired with masking schemes to ensure that the variance of the noise is sufficiently high.

There are additional aspects which depending on the prior knowledge of the attacker and type of attack can significantly increase the attack complexity for higher orders. If the attacker does not know which sample points need to be combined for a successful attack, the search for the points of interest adds another difficulty to the attack procedure that can grow exponentially with the order. Since testing every possible combination of the sample points become quickly unfeasible for large d, more sophisticated methods have been proposed to accelerate this attack step [RGV12, DS16]. Nevertheless, the search for point of interests is still a necessary obstacle for *multivariate* higher-order attacks which require the combination of different leakage samples as it is customary in the attack of masked software implementations. On the other hand, *univariate* higher-order attacks are not affected by this issue as each leakage sample is considered separately. This is a particularly relevant for our hardware-based masked designs in which all shares are processed in parallel and thus a combination of different leakage samples is not required. Instead, each leakage sample is preprocessed as

$$Y = (X - \mu_X)^{d+1}\,, \tag{2.4}$$

making the search step unnecessary. In this thesis, we mostly consider univariate attacks due to the parallel processing of our masked hardware architectures. For the two cases which require a multivariate evaluation, we do not apply a sophisticated search approach and instead rely on a simple shift of the traces by a constant factor which is sufficient for our applications.

Part II

Evaluation Methodologies for Side-Channel Countermeasures

Chapter 3

Background: Evaluation Methodologies

In this chapter, we give background information relevant to the research contribution of this part. Fundamental statistical concepts including statistical moments and their influence on distributions are introduced. Additionally, we give a brief overview of the principle of iterative computation for higher-order statistical moments. The latter description is based in parts on [SMG16c].

Contents of this Chapter

3.1 Statistical Background 19
3.2 Iterative Computation 21

3.1 Statistical Background

For an easier understanding of the results presented in this part, we briefly introduce the main statistical concepts used in all evaluation chapters. To this end, we describe the different types of statistical moments and show the exemplary normal distribution.

3.1.1 Statistical Moments

Statistical moments can be used to describe the shape of a distribution. There are different types of moments which we define in the following.

Definition 3.1.1. The d-order raw moment of the random variable X is given by

$$M_d = \mathsf{E}\left(X^d\right). \tag{3.1}$$

Definition 3.1.2. The d-order central moment of the random variable X is given by

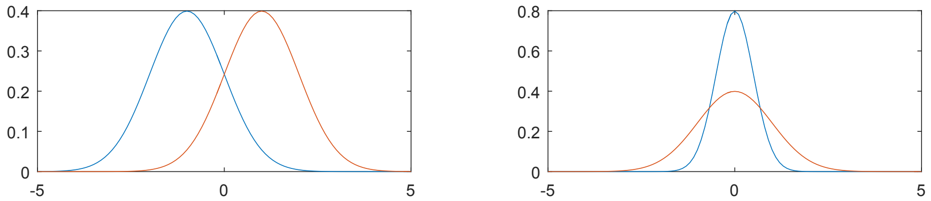
$$CM_d = \mathsf{E}\left((X-\mu)^d\right) \tag{3.2}$$

where μ denotes the mean of X.

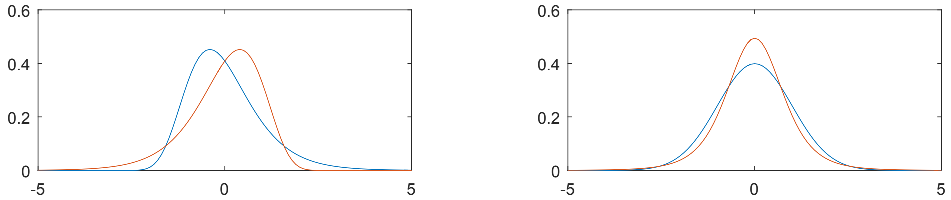
Definition 3.1.3. The d-order standardized moment of the random variable X is given by

$$SM_d = \mathsf{E}\left(\left(\frac{X-\mu}{\sigma}\right)^d\right) \tag{3.3}$$

where μ and σ denote the mean and standard deviation of X.



(a) Mean. (b) Variance.



(c) Skewness. (d) Kurtosis.

Figure 3.1: The influence of the first four statistical moments on the location, spread, and shape of a distribution.

M_1 denotes the first-order *raw* statistical moment which is the mean $\mu = \mathsf{E}(X)$ of the variable. CM_2 denotes the second-order *central* statistical moment which is the variance $\sigma^2 = \mathsf{E}\left((X-\mu)^2\right)$ with $\sigma = \sqrt{CM_2}$ being the standard deviation. SM_3 denotes the third-order *standardized* moment which is the skewness $\gamma_1 = \mathsf{E}\left(\left(\frac{X-\mu}{\sigma}\right)^3\right)$. Furthermore, the fourth-order standardized moment SM_4 denotes the kurtosis $\beta_2 = \mathsf{E}\left(\left(\frac{X-\mu}{\sigma}\right)^4\right)$. In the following, we refer to $\mu, \sigma^2, \gamma_1, \beta_2$ only as the first four statistical moments and do not specifically note their type due to their importance for our evaluation methodologies. For any other raw, central, or standardized moment we explicitly state their type to avoid confusion.

These four moments are particular important as changes to these moments directly affect the location (mean), spread (variance), and shape (skewness, kurtosis) of the distribution. We briefly visualize the influence of the first four statistical moments on a distribution. Each part of Figure 3.1 depicts the influence of one of the four moments individually. By changing the mean, the whole distribution is shifted by a fixed offset. The variance affects the spread of the distribution, meaning that higher variance values lead to wider distributions. For symmetric distributions the skewness is zero, meaning that a non-zero skewness indicates a form of asymmetry resulting in distributions which tend to lean to the right or left. Finally, the kurtosis provides information about the sharpness of a distribution as shown in Figure 3.1(d).

Mixed moments exist for multiple variables and contain information about the joint distribution of these variables. An example is the covariance between two variables X, Y defined as

$$\operatorname{cov}(X,Y) = \mathsf{E}\left((X-\mathsf{E}(X))(Y-\mathsf{E}(Y))\right). \tag{3.4}$$

In general, we define mixed moments as the centered product between multiple variables. However, these are of lesser importance to the evaluation process of our masked hardware designs, since we deal with univariate leakages in most of our experiments.

In side-channel analysis, we derive the statistical moments, which are required for an evaluation of a specific order, from a limited number of actual measurements. Therefore, the resulting moments are not the ideal statistical moments described above, but only an estimation whose accuracy depends on the number of samples. These sample moments can be biased and require particular care during computation, e.g., the unbiased sample variance s^2 is defined as

$$s^2 = \frac{1}{1-n} \sum_{i=1}^{n} \left(X - \overline{X} \right)^2 \tag{3.5}$$

where $\overline{X}$ denotes the sample mean and n the number of samples. However, for most side-channel applications n is a relatively large number which makes the difference between the biased and unbiased estimator negligible for our evaluations. Nevertheless, all of our results can be easily adjusted to incorporate the unbiased estimator.

3.1.2 Normal Distribution

The normal (or Gaussian) distribution is commonly-used in side-channel analysis as a model for the measurements. This is mainly due to the measurement noise which is often roughly normally distributed and thus the measurements for a fixed input follow a normal distribution. It can be parameterized with the first two statistical moments μ and σ^2 and the PDF of the normal distribution is

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3.6}$$

which returns the relative likelihood of the random variable X to be a specific value x. The distribution is symmetric (i.e., the skewness is zero) and has a constant kurtosis of three. Figure 3.1(a) depicts two exemplary normal distributions with different means. Due to its efficiency and effectiveness, the normal distribution is often chosen for scenarios which only require the mean and variance. However, during our experiments we show that higher-order moments can also contain information and should not be neglected during the evaluation. For these cases, we propose to use other distributions which consider more than the first two statistical moments.

3.2 Iterative Computation

For a given set of samples, an *one-pass* algorithm processes each sample of the set only once to compute the desired result. In the side-channel evaluation scenario, this relates to loading each measurement trace only once to perform the evaluation which for our methodologies often involves the computation of various statistical moments. While the raw moments can be straightforwardly computed with one pass, central and standardized moments require more sophisticated algorithms as their preprocessing includes precomputed moments, i.e., central moments require μ and standardizes moments require μ, σ. Therefore, a naive implementation to compute the standardized moments would first compute the sample mean and variance in two passes and perform a third pass over the measurements to derive the desired moments. Since this approach is inefficient, different solutions to enable one-pass computation for these cases have been proposed. We shortly recall existing solutions based on *raw* and *central* sums.

Definition 3.2.1. Given a set of n sample points x_i, the d-order raw sum S_d of the set is defined as

$$S_d = \sum_{i=1}^{n} x_i^d. \tag{3.7}$$

Obviously, these sums are easily computed in a one-pass fashion and can be used to derive the raw moments as $M_d = \frac{S_d}{n}$. Nevertheless, we shortly recall the update function for the raw sums from [Péb08]. Suppose that $M_{1,\mathcal{Q}_1}$ (resp. $M_{1,\mathcal{Q}_2}$) denotes the first raw moment (sample mean) of the given set $\mathcal{Q}_1$ (resp. $\mathcal{Q}_2$) with cardinality $n_1 = |\mathcal{Q}_1|$ and $n_2 = |\mathcal{Q}_2|$. $M_{1,\mathcal{Q}}$ as the first raw moment of $\mathcal{Q} = \mathcal{Q}_1 \cup\ \mathcal{Q}_2$ can be written as [Péb08]

$$M_{1,\mathcal{Q}} = \frac{n_1\, M_{1,\mathcal{Q}_1} + n_2\, M_{1,\mathcal{Q}_2}}{n}, \tag{3.8}$$

with $n = n_1 + n_2$ as the cardinality of $\mathcal{Q}$. Furthermore, multiple specific raw moments can be combined to compute central and standardized moments, e.g., $CM_2 = M_2 - M_1^2$. As practically shown later in Chapter 4, this approach suffers from numerical instabilities due to the size of the raw sums and cannot be easily used for the computation of higher-order moments of sets with a large number of samples, e.g., for higher-order side-channel evaluations.

Definition 3.2.2. Given a set of n sample points x_i, the d-order central sum CS_d of the set is defined as

$$CS_d = \sum_{i=1}^{n} (x_i - \mu)^d \tag{3.9}$$

where μ denotes the mean of the set.

The central sums are related to the central moments as $CM_d = \frac{CS_d}{n}$. Due to the subtraction of the mean from the sample points, the resulting central sums of a given order are significantly smaller than raw sums of equal order for a non-zero mean. This helps to minimize the aforementioned numerical instabilities and enables the efficient computation of higher-order moments. However, in contrast to the raw sums, it is not trivial to compute the central sums of arbitrary orders in one pass. Such an update formula can be written for the central sum $CS_{d,\mathcal{Q}}$ at any arbitrary order $d > 1$ as [Péb08]

$$\begin{aligned} CS_{d,\mathcal{Q}} = CS_{d,\mathcal{Q}_1} + CS_{d,\mathcal{Q}_2} + \sum_{p=1}^{d-2} \binom{d}{p} \left[\left(\frac{-n_2}{n}\right)^p CS_{d-p,\mathcal{Q}_1} + \left(\frac{n_1}{n}\right)^p CS_{d-p,\mathcal{Q}_2} \right] \Delta^p \\ + \left(\frac{n_1\, n_2}{n}\, \Delta\right)^d \left[\left(\frac{1}{n_2}\right)^{d-1} - \left(\frac{-1}{n_1}\right)^{d-1} \right], \end{aligned} \tag{3.10}$$

with $\Delta = M_{1,\mathcal{Q}_2} - M_{1,\mathcal{Q}_1}$. It is noteworthy that the calculation of $CS_{d,\mathcal{Q}}$ additionally requires $CS_{p,\mathcal{Q}_1}$ and $CS_{p,\mathcal{Q}_2}$ for $1 < p \leq d$.

Furthermore, we define two terms *iterative* and *incremental* which are frequently used for the rest of the dissertation. Suppose that after finishing all the required processes on the set $\mathcal{Q}$, a new sample point y is added to the set $\mathcal{Q}' = \mathcal{Q} \cup \{y\}$. *Incremental* formulas allow updating the previously computed terms by only processing the new trace y. In addition to that, we suppose that the set $\mathcal{Q}$ is divided into two subsets as $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$, and each subset is independently processed using the given incremental formulas. *Iterative* formulas enable the combination of results computed over each group $\mathcal{Q}_1$ and $\mathcal{Q}_2$ to derive the result of the full trace pool $\mathcal{Q}$.

Chapter 4

Leakage Assessment Methodology

In this chapter, we present our contributions related to test-based evaluation methodologies for side-channel countermeasures based on [SM15a, SM16]. One of such evaluation methodologies is the Welch's t-test, which is being put forward by Cryptography Research Inc., and is able to relax the dependency between the evaluations and the device's underlying architecture. We deeply study the theoretical background of the test's different flavors, and present a roadmap which can be followed by the evaluation labs to efficiently and correctly conduct the tests. More precisely, we express a stable, robust and efficient way to perform the tests at higher orders. Further, we extend the test to multivariate settings, and provide details on how to efficiently and rapidly carry out such a multivariate higher-order test. Including a suggested methodology to collect the traces for these tests, we present practical case studies where different types of t-tests can exhibit the leakage of supposedly secure designs.

Contents of this Chapter

4.1 Introduction **25**
4.2 Background **27**
4.3 Methodology **28**
4.4 Efficient Computation **31**
4.5 Multivariate Evaluation **34**
4.6 Case Studies **37**
4.7 Conclusion **43**

4.1 Introduction

With respect to common criteria evaluations – defined and used by governing bodies like Agence nationale de la sécurité des systèmes d'information (ANSSI) and Bundesamt für Sicherheit in der Informationstechnik (BSI) – the evaluation labs need to *practically* examine the feasibility of the state-of-the-art attacks conducted on the Device Under Test (DUT). The examples include but not restricted to the classical Differential Power Analysis (DPA) [KJJ99], CPA [BCO04], and MIA [GBTP08]. To cover the most possible cases a large range of intermediate values as well as hypothetical (power) models should be examined to assess the possibility of the key recovery. This methodology is becoming more challenging as the number and types of known side-channel attacks are steadily increasing. Trivially, this time-consuming procedure cannot be comprehensive even if a large number of intermediate values and models in addition to several know attacks are examined. In fact, the selection of the hypothetical model is not simple and strongly depends on the expertise of the evaluation labs' experts. If the models were poorly chosen and as a result none of the key-recovery attacks succeeded, the evaluation lab would issue a favorable evaluation report even though the DUT might be vulnerable to an attack with a more advanced and complex model. This motivates the need for an evaluation procedure which avoids being dependent on attack(s), intermediate value(s), and hypothetical model(s).

On one hand, two information-theoretic tests [CG11, CCG10] are known which evaluate the leakage distributions either in a continuous or discrete form. These approaches are based on the mutual information and need to estimate the probability distribution of the leakages. This adds other parameter(s) to the test with respect to the type of the employed density estimation technique, e.g., kernel or histogram and their corresponding parameters. Moreover, they cannot yet focus on a certain statistical order of the leakages. This becomes problematic when e.g., the first-order security of a masking countermeasure is expected to be assessed. On the other hand, two leakage assessment methodologies (*specific* and *non-specific* t-tests) based on the Student's t-distribution have been proposed (at the aforementioned workshop [GJJR11]) with the goal to detect any type of leakage at a certain order. A comparative study of these three test vectors is presented in [MOBW13], where the performance of specific t-tests (only at the first order) is compared to that of other mutual information-based tests.

In general, the *non-specific* t-test examines the leakage of the DUT without performing an actual attack, and is in addition independent of its underlying architecture. The test gives a level of confidence to conclude that the DUT has an exploitable leakage. It indeed provides no information about the easiness/hardness of an attack which can exploit the leakage, nor about an appropriate intermediate value and the hypothetical model. However, it can easily and rapidly report that the DUT fails to provide the desired security level, e.g., due to a mistake in the design engineering or a flaw in the countermeasure [BGG+14].

4.1.1 Contribution

The Welch's t-test has been used in a couple of research works [BGG+14, LMW14, BGN+14b, SMG15b, SMMG15b, MH15, SMMG15a, WMG15] to investigate the efficiency of the proposed countermeasures, but without extensively expressing the challenges of the test procedure. This document aims at putting light on a path for e.g., evaluation labs, on how to examine the leakage of the DUT at any order with minimal effort and without any dependency to a hypothetical model. Our goal in this work is to cover the following points:

(1) We try to explain the underlying statistical concept of such a test by a (hopefully) more understandable terminology.

(2) In the seminal paper by Goodwill et al. [GJJR11] it has been shown how to conduct the test at the first order, i.e., how to investigate the first-order leakage of the DUT. The authors also shortly stated that the traces can be preprocessed to run the same test at higher orders. Here we point out the issues one may face to run such a test at higher orders, and provide appropriate solutions accordingly. As a motivating point we should refer to [MOBW13], where the t-test is supposed to be able to be performed at only the first order.

(3) More importantly, we extend the test to cover multivariate leakages and express the necessary formulations in detail allowing us to efficiently conduct t-tests at any order and any variate.

(4) In order to evaluate the countermeasures (mainly those based on masking at high orders) several million traces might be required (e.g., see [LMW14, BGN$^+$14b]). Hence, we express the procedures which allow conducting the tests by means of multi-core CPUs in a parallelized way.

(5) We give details of how to design appropriate frameworks to host the DUT for such tests, including both software and hardware platforms. Particularly we consider a microcontroller as well as an FPGA (SASEBO) for this purpose.

(6) Depending on the underlying application and platform, the speed of the measurement is a bottleneck which hinders the collection of several million measurements. Due to this reason, the evaluation labs are usually restricted (commonly by common criteria) to measure not more than one million traces from any DUT. We also demonstrate a procedure to accelerate the measurement process allowing the collection of e.g., millions of traces per hour.

(7) We also show two practical case studies, where the univariate as well as bivariate t-tests show the leakage of designs expected to be secure.

4.2 Background

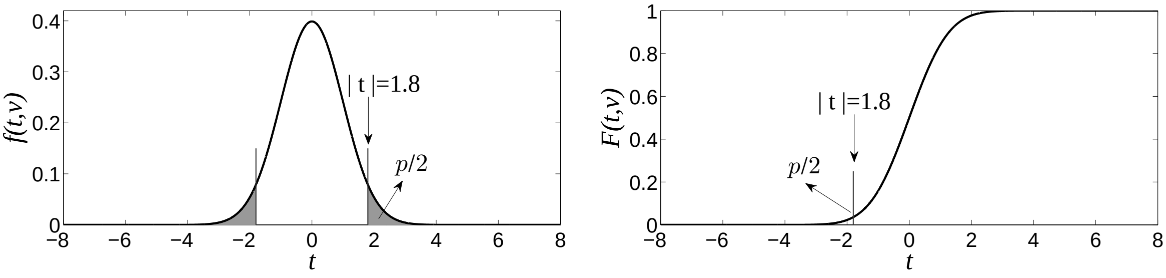
A fundamental question in many scientific fields is whether two sets of data are significantly different from each other. The most common approach to answer such a question is Welch's t-test in which the test statistic follows a Student's t distribution. The aim of a t-test is to provide a quantitative value as a probability that the mean μ of two sets are different. In other words, a t-test gives a probability to examine the validity of the null hypothesis *as the samples in both sets were drawn from the same population*, i.e., the two sets are not distinguishable.

Hence, let $\mathcal{Q}_0$ and $\mathcal{Q}_1$ indicate two sets which are under the test. Let also μ_0 (resp. μ_1) and ${\sigma_0}^2$ (resp. ${\sigma_1}^2$) stand for the sample mean and sample variance of the set $\mathcal{Q}_0$ (resp. $\mathcal{Q}_1$), and n_0 and n_1 the cardinality of each set. The t-test statistic and the degree of freedom v are computed as

$$\mathrm{t} = \frac{\mu_0 - \mu_1}{\sqrt{\frac{{\sigma_0}^2}{n_0} + \frac{{\sigma_1}^2}{n_1}}}, \qquad \mathrm{v} = \frac{\left(\frac{{\sigma_0}^2}{n_0} + \frac{{\sigma_1}^2}{n_1}\right)^2}{\frac{\left(\frac{{\sigma_0}^2}{n_0}\right)^2}{n_0-1} + \frac{\left(\frac{{\sigma_1}^2}{n_1}\right)^2}{n_1-1}}. \tag{4.1}$$

In cases, where $\sigma_0 \approx \sigma_1$ and $n_0 \approx n_1$, the degree of freedom can be estimated by $\mathrm{v} \approx n_0 + n_1 = n$. As the final step, we estimate the probability to accept the null hypothesis by means of Student's t distribution density function. In other words, based on the degree of freedom v the Student's t distribution function is drawn

$$f(t,v) = \frac{\Gamma(\frac{v+1}{2})}{\sqrt{\pi v}\,\Gamma(\frac{v}{2})}\left(1 + \frac{t^2}{v}\right)^{-\frac{v+1}{2}},$$



(a) probability density function (b) cumulative distribution function

Figure 4.1: Student's t distribution functions and two-tailed Welch's t-test (examples for $v = 10,000$).

where $\Gamma(.)$ denotes the gamma function. Based on the two-tailed Welch's t-test the desired probability is calculated as

$$p = 2 \int_{|\mathrm{t}|}^{\infty} f(t, \mathrm{v})\, dt.$$

Figure 4.1(a) represents a graphical view of such a test.

As an alternative, we can make use of the corresponding cumulative distribution function

$$F(t,v) = \frac{1}{2} + t\,\Gamma\left(\frac{v+1}{2}\right) \frac{{}_2F_1\left(\frac{1}{2}, \frac{v+1}{2}; \frac{3}{2}; -\frac{x^2}{v}\right)}{\sqrt{\pi v}\,\Gamma\left(\frac{v}{2}\right)},$$

with ${}_2F_1(.,.;.;.)$ the hypergeometric function. Hence, the result of the t-test can be estimated as

$$p = 2\,F(-|\mathrm{t}|, \mathrm{v}).$$

For a graphical view see Figure 4.1(b). Note that such a function is available among the MATLAB embedded functions as `tcdf(·,·)` and for R as `qt(·,·)`.

Hence, small p values (alternatively big t values) give evidence to reject the null hypothesis and conclude that the sets were drawn from different populations. For the sake of simplicity, usually a threshold $|\mathrm{t}| > 4.5$ is defined to reject the null hypothesis without considering the degree of freedom and the aforementioned cumulative distribution function. This intuition is based on the fact that $p = 2\,F(-4.5, v > 1000) < 0.00001$ which leads to a confidence of > 0.99999 to reject the null hypothesis.

4.3 Methodology

Suppose that in a side-channel evaluation process, with respect to n queries with associated data (e.g., plaintext or ciphertext) $D_{i\in\{1,\ldots,n\}}$, n side-channel measurements (so-called traces) are collected while the device under test operates with a secret key that is kept constant. Let us denote each trace by $T_{i\in\{1,\ldots,n\}}$ containing m sample points $\{t_i^{(1)}, \ldots, t_i^{(m)}\}$.

As a straightforward evaluation process, the traces are categorized into two sets $\mathcal{Q}_0$ and $\mathcal{Q}_1$ and the test is conducted at each sample point $\{1, \ldots, m\}$ separately. In other words, the test is performed in a *univariate* fashion. At this step such a categorization is done by means of an intermediate value corresponding to the associated data D. Since the underlying process is an evaluation procedure, the secret key is known and all the intermediate values can be computed. Based on the concept of the classical DPA [KJJ99], a bit of an intermediate value (e.g., an Sbox output bit at the first cipher round) is selected to be used in the categorization.

$$\mathcal{Q}_0 = \{T_i \,|\, \text{target bit}(D_i) = 0\}, \qquad \mathcal{Q}_1 = \{T_i \,|\, \text{target bit}(D_i) = 1\}.$$

If the corresponding t-test reports that with a high confidence the two trace groups (at certain sample points) are distinguishable from each other, it is concluded that the corresponding DPA attack is – most likely – able to recover the secret key.

Such a test (so-called *specific* t-test) is not restricted to only single-bit scenarios. For instance, an 8-bit intermediate value (e.g., an Sbox output byte) can be used to categorize the traces as

$$\mathcal{Q}_0 = \{T_i \,|\, \text{target byte}(D_i) = \text{x}\}, \qquad \mathcal{Q}_1 = \{T_i \,|\, \text{target byte}(D_i) \neq \text{x}\}.$$

In this case, a particular value for x should be selected prior to the test. Therefore, in case of an 8-bit target intermediate value 256 specific t-tests can be performed. It should be noted that in such tests, n_0 and n_1 (as the cardinality of $\mathcal{Q}_0$ and $\mathcal{Q}_1$) would be significantly different if the associated data D were drawn randomly. Hence, the accuracy of the estimated (sample) means (μ_0, μ_1) as well as variances (${\sigma_0}^2$, ${\sigma_1}^2$) would not be the same. However, this should not – in general – cause any issue as the two-tailed Welch's t-test covers such a case.

Therefore, the evaluation can be performed by many different intermediate values. For example, in case of an AES-128 encryption engine by considering the AddRoundKey, SubBytes, ShiftRows, and MixColumns outputs, 4 × 128 bit-wise tests and 4 × 16 × 256 byte-wise tests (only at the first cipher round) can be conducted. This already excludes the XOR result between the intermediate values, which depending on the underlying architecture of the DUT (e.g., a serialized architecture) may lead to potential leaking sources. Therefore, such tests suffer from the same weakness as state-of-the-art attacks since both require to examine many intermediate values and models, which prevents a comprehensive evaluation.

To cope with this imperfection a *non-specific* t-test can be performed, which avoids being dependent on any intermediate value or a model. In such a test the associated data should follow a certain procedure during the trace collection. More precisely a fixed associated data D is preselected, and the DUT is fed by D or by a random source in a non-deterministic and randomly-interleaved fashion. As a more clear explanation suppose that before each measurement a coin is flipped, and accordingly D or a fresh-randomly selected data is given to the DUT. The corresponding t-test is performed by categorizing the traces based on the associated data (D or random). Hence, such a test is also called *fixed vs. random* t-test.

The randomly-interleaved procedure is unavoidable; otherwise the test may issue a false-positive result on the vulnerability of the DUT. It is mainly due to the fact that the internal state of the DUT at the start of each query should be also non-deterministic. As an example, if the traces with associated data D are collected consecutively, the DUT internal state is always the same prior to each measurement with D. As another example, if the traces with random associated data and D are collected one after each other (e.g., D_i being random for even i and D for odd i), the DUT internal state is always the same prior to each measurement with random associated data.

In order to explain the concept behind the non-specific t-test, assume a specific t-test based on a single-bit intermediate variable w of the underlying process of the DUT and the corresponding sample point j where the leakage associated to w is measured. Further, let us denote the estimated means of the leakage traces at sample point j by $\mu_{w=0}$ and $\mu_{w=1}$, i.e., those applied in the specific t-test. If these two means are **largely enough** different from each other, each of them is also distinguishable from the overall mean μ ($\approx \frac{\mu_{w=0} + \mu_{w=1}}{2}$ supposing $n_0 \approx n_1$).

From another perspective, consider two non-specific t-tests with the fixed associated data $\mathrm{D}_{w=0}$ and $\mathrm{D}_{w=1}$, where $\mathrm{D}_{w=0}$ leads to the intermediate value $w = 0$ (respectively for $\mathrm{D}_{w=1}$). Also, suppose that in each of these two tests $\mathcal{Q}_0$ corresponds to the traces with the fixed associated data and $\mathcal{Q}_1$ to those with random. Hence, in the non-specific test with $\mathrm{D}_{w=0}$, the estimated mean μ_0 at sample point j is close to $\mu_{w=0}$ (respectively to $\mu_{w=1}$ in the test with $\mathrm{D}_{w=1}$). But in both tests the estimated mean μ_1 (of $\mathcal{Q}_1$) is close to μ (defined above). Therefore, in both tests the statistic ($\mathrm{t}^{non-spec.}$) is smaller than that of the specific test ($\mathrm{t}^{spec.}$) since $\mu_{w=0} < \mu < \mu_{w=1}$ (or respectively $\mu_{w=1} < \mu < \mu_{w=0}$). However, even supposing $n_0 \approx n_1$ it **cannot** be concluded that

$$|\mathrm{t}^{non-spec.}| = |\mathrm{t}^{spec.}|/2$$

since the estimated overall variance at sample point j (which is that of $\mathcal{Q}_1$ in both non-specific tests) is

$$\sigma_1{}^2 = \frac{(\sigma_{w=0})^2 + (\sigma_{w=1})^2}{2} + \left(\frac{\mu_{w=0} - \mu_{w=1}}{2}\right)^2 \neq (\sigma_{w=0/1})^2,$$

assuming $n_0 \approx n_1$.

As a result if a non-specific t-test reports a detectable leakage, the specific one results in the same conclusion but with a higher confidence. Although any intermediate value (either bit-wise or at larger scales) as well as the combination between different intermediate values are covered by the non-specific t-test, the negative result (i.e., no detectable leakage) cannot be concluded from a single non-specific test due to its dependency to the selected fixed associated data D. In other words, it may happen that a non-specific t-test by a certain D reports no exploitable leakage, but the same test using another D leads to the opposite conclusion. Hence, it is recommended to repeat a non-specific test with a couple of different D to avoid a false-positive conclusion on resistance of the DUT.

The non-specific t-test can also be performed by a set of particular associated data $\mathcal{D}$ instead of a unique D. The associated data in $\mathcal{D}$ are selected in such a way that all of them lead to a certain intermediate value. For example, a set of plaintexts which cause half of the cipher state at a particular cipher round to be constant. In this case $\mathcal{Q}_0$ refers to the traces with associated data – randomly – selected from $\mathcal{D}$ (respectively $\mathcal{Q}_1$ to the traces with random associated data). Such a non-specific t-test is also known as the *semi-fixed vs. random* test [CDG+13], and is particularly useful where the test with a unique D leads to a false-positive result on the vulnerability of the DUT. We express the use cases of each test in more details in Section 4.6.

4.3.1 Order of the Test

Recalling the definition of first-order resistance, the estimated means of leakages associated to the intermediate values of the DUT should not be distinguishable from each other (i.e., the concept behind the Welch's t-test). Otherwise, if such an intermediate value is sensitive and predictable knowing the associated data D (e.g., the output of an Sbox at the first cipher round) a corresponding first-order DPA/CPA attack is expected to be feasible. It can also be extended to the higher orders by following the definition of univariate higher-order attacks [MM12]. To do so (as also stated in [GJJR11]) the collected traces need to be preprocessed. For example, for a second-order evaluation each trace – at each sample point independently – should be mean-free squared prior to the t-test. Here we formalize this process slightly differently as follows.

In a first-order univariate t-test, for each set ($\mathcal{Q}_0$ or $\mathcal{Q}_1$) the mean (M_1) is estimated. For a second-order univariate test the mean of the mean-free squared traces $Y = (X - \mu)^2$ is actually the variance (CM_2) of the original traces. Respectively, in a third and higher ($d > 2$) order test the standardized moment SM_d is the estimated mean of the preprocessed traces. Therefore, the higher-order tests can be conducted by employing the corresponding estimated (central or standardized) moments instead of the means. The remaining point is how to estimate the variance of the preprocessed traces for higher-order tests. We deal with this issue in Section 4.4.2 and explain the corresponding details.

As stated, all the above given expressions are with respect to univariate evaluations, where the traces at each sample point are independently processed. For a bivariate (respectively multivariate) higher-order test different sample points of each trace should be first combined prior to the t-test, e.g., by centered product at the second order. A more formal definition of these cases is given in Section 4.5.

4.4 Efficient Computation

As stated in the previous section, the first order t-test requires the estimation of two parameters (sample mean μ and sample variance σ^2) for each set $\mathcal{Q}_0$ and $\mathcal{Q}_1$. This can lead to problems concerning the efficiency of the computations and the accuracy of the estimations. In the following we address most of these problems and propose a reasonable solution for each of them. For simplicity, we omit to mention the sets $\mathcal{Q}_0$ and $\mathcal{Q}_1$ (and the corresponding indices for the means and variances). All the following expressions are based on focusing on one of these sets, which should be repeated on the other set to complete the required computations of a t-test. Unless otherwise stated, we focus on a univariate scenario. Hence, the given expressions should be repeated at each sample point separately.

Using the basic definitions given in Section 3.1, it is possible to compute the first raw and second central moments (M_1 and CM_2) for a first order t-test. However, the resulting algorithm is inefficient as it requires to process the whole trace pool (a single point) twice to estimate CM_2 since it requires M_1 during the computation.

An alternative would be to use the displacement law to derive CM_2 from the first two raw moments as

$$CM_2 = \mathsf{E}(X^2) - \mathsf{E}(X)^2 = M_2 - {M_1}^2. \tag{4.2}$$

Whereas it results in a one-pass algorithm, it is still not the optimal choice as it may be numerically unstable [Hig02]. During the computation of the raw moments the intermediate values tend to become very large which can lead to a loss in accuracy. Further, M_2 and ${M_1}^2$ can be large values, and the result of $M_2 - {M_1}^2$ can also lead to a significant accuracy loss due to the limited fraction significand of floating point formats (e.g., IEEE 754).

In the following we present a way to compute the two required parameters for the t-test at any order in one pass and with proper accuracy. This is achieved by using an incremental algorithm to update the *central sums* from which the needed parameters are derived.

4.4.1 Incremental One-Pass Computation of All Moments

The basic idea of an incremental algorithm for side-channel evaluation is to update the intermediate results for each new trace added to the trace pool. This has the advantage that the computation can be run in parallel to the measurement phase. In other words, it is not necessary to collect all the traces, estimate the mean and then estimate the variance. Since the evaluation can be stopped as soon as the t-value surpasses the threshold, this helps to reduce the evaluation time even further. In the following we show how to efficiently derive the standardized moments and in turn compute the parameters for univariate t-tests at arbitrary orders. To this end, we first recall the incremental version of the iterative update functions from Section 3.2.

Suppose that $M_{1,\mathcal{Q}}$ denotes the first raw moment (sample mean) of the given set $\mathcal{Q}$. With y as a new trace to the set, the first raw moment of the enlarged set $\mathcal{Q}' = \mathcal{Q} \cup \{y\}$ can be updated as

$$M_{1,\mathcal{Q}'} = M_{1,\mathcal{Q}} + \frac{\Delta}{n},$$

where $\Delta = y - M_{1,\mathcal{Q}}$, and n the cardinality of $\mathcal{Q}'$. Note that $\mathcal{Q}$ and $M_{1,\mathcal{Q}}$ are initialized with $\emptyset$ and respectively zero.

Following the same definitions, the formula to update CS_d can be written as [Péb08]

$$CS_{d,\mathcal{Q}'} = CS_{d,\mathcal{Q}} + \sum_{k=1}^{d-2} \binom{d}{k} CS_{d-k,\mathcal{Q}} \left(\frac{-\Delta}{n}\right)^k + \left(\frac{n-1}{n}\Delta\right)^d \left[1 - \left(\frac{-1}{n-1}\right)^{d-1}\right], \tag{4.3}$$

where Δ is still the same as defined above.

Based on these formulas the first raw and all central moments can be computed efficiently in one pass. Furthermore, since the intermediate results of the central sums are mean free, they do not become significantly large which helps to prevent the numerical instabilities. Since the standardized moments are the central moments which are normalized by the variance, they can be easily derived from the central moments as

$$SM_d = \frac{1}{n} \sum_i \left(\frac{x_i - \mu}{\sigma}\right)^d = \frac{CM_d}{\left(\sqrt{CM_2}\right)^d}. \tag{4.4}$$

Therefore, the first parameter of the t-test (mean of the preprocessed data) at any order can be efficiently and precisely estimated. Below we express how to derive the second parameter for such tests at any order.

4.4.2 Variance of Preprocessed Traces

A t-test at higher orders operates on preprocessed traces. In particular, it requires to estimate the variance of the preprocessed traces. Such a variance does in general not directly correspond to a central or standardized moment of the original traces. Below we present how to derive such a variance at any order from the central and standardized moments.

Equation (4.2) shows how to obtain the variance given only the first two raw moments. We extend this approach to derive the variance of the preprocessed traces. In case of the second order, the traces are mean-free squared, i.e., $Y = (X - \mu)^2$. The variance of Y is estimated as

$$\begin{aligned}
{\sigma_Y}^2 &= \frac{1}{n}\sum\Big((x-\mu)^2 - \frac{1}{n}\sum(x-\mu)^2\Big)^2 = \frac{1}{n}\sum\Big((x-\mu)^2 - CM_2\Big)^2 \\
&= \frac{1}{n}\sum(x-\mu)^4 - \frac{2}{n}CM_2\sum(x-\mu)^2 + {CM_2}^2 \\
&= CM_4 - {CM_2}^2. \qquad (4.5)
\end{aligned}$$

Therefore, the sample variance of the mean-free squared traces (required for a second-order t-test) can be efficiently derived from the central moments CM_4 and CM_2. Note that the values processed by the above equations (CM_4 and CM_2) are already centered hence avoiding the instability issue addressed in Section 4.4. For the cases at the third order, the traces are additionally standardized, i.e., $Z = (\frac{X-\mu}{s})^3$. The variance of Z can be written as

$$\begin{aligned}
{\sigma_Z}^2 &= \frac{1}{n}\sum\Big((\frac{x-\mu}{s})^3 - \frac{1}{n}\sum(\frac{x-\mu}{s})^3\Big)^2 = \frac{1}{n}\sum\Big((\frac{x-\mu}{s})^3 - SM_3\Big)^2 \\
&= \frac{1}{n}\sum(\frac{x-\mu}{s})^6 - \frac{2}{n}SM_3\sum(\frac{x-\mu}{s})^3 + {SM_3}^2 \\
&= SM_6 - {SM_3}^2 = \frac{CM_6 - {CM_3}^2}{{CM_2}^3}. \qquad (4.6)
\end{aligned}$$

Since the tests at third and higher orders use standardized traces, it is possible to generalize Equation (4.6) for the variance of the preprocessed traces at any order $d > 2$ as

$$SM_{2d} - {SM_d}^2 \;=\; \frac{CM_{2d} - {CM_d}^2}{{CM_2}^d}. \qquad (4.7)$$

Therefore, a t-test at order d requires to estimate the central moments up to order $2d$. With the above given formulas it is now possible to extend the t-test to any arbitrary order as we can estimate the corresponding required first and second parameters efficiently. In addition, most of the numerical problems are eliminated in this approach. The required formulas for all parameters of the tests up to the fifth order are provided in Appendix 12. We also included the formulas when the first and second parameters of the tests (up to the fifth order) are derived from raw moments.

In order to give an overview on the accuracy of different ways to compute the parameters for the t-tests, we ran an experiment with 100 million simulated traces with $\sim \mathcal{N}(100, 25)$, which fits to a practical case where the traces (obtained from an oscilloscope) are signed 8-bit integers. We computed the second parameter for t-tests using i) three-pass algorithm, ii) the raw moments, and iii) our proposed method. Note that in the three-pass algorithm first the mean μ is estimated. Then, having μ the traces are processed again to estimate all required central and standardized moments, and finally having all moments the traces are preprocessed (with respect to the desired order) and the variances (of the preprocessed traces) are estimated. The corresponding results are shown in Table 4.1. In terms of accuracy, our method matches the three-pass algorithm. The raw moments approach suffers from severe numerical instabilities, especially at higher orders where the variance of the preprocessed traces becomes negative.

Table 4.1: Comparison of the accuracy of different methods to compute the second parameter of the t-tests (100 million simulated traces $\sim \mathcal{N}(100, 25)$).

	1st order	*2nd order*	*3rd order*	*4th order*	*5th order*
Three Pass	25.08399	1258.18874	15.00039	96.08342	947.25523
Raw Moments	25.08399	1258.14132	14.49282	-1160.83799	-1939218.83401
Our Method	25.08399	1258.18874	15.00039	96.08342	947.25523

4.4.3 Parallel Computation

Depending on the data complexity of the measurements, it is sometimes favorable to parallelize the computation in order to reduce the time complexity. To this end, a straightforward approach is to utilize a multi-core architecture (a CPU cluster) which computes the necessary central sums for multiple sample points in parallel. This can be achieved easily as the computations on different sample points are completely independent of each other. Consequently, there is no communication overhead between the threads. This approach is beneficial in most measurement scenarios and enables an extremely fast evaluation depending on the number of available CPU cores as well as the number of sample points in each trace. As an example, we are able to calculate all the necessary parameters of five non-specific t-tests (at first to fifth orders) on $100,000,000$ traces (each with $3,000$ sample points) in 9 hours using two Intel Xeon X5670 CPUs @ 2.93 GHz, i.e., 24 hyper-threading cores.

A different approach can be preferred if the number of points of interest is very low. In this scenario, suppose that the trace collection is already finished and the t-tests are expected to be performed on a small number of sample points of a large number of traces. The aforementioned approach for parallel computing might not be the most efficient way as the degree of parallelization is bounded by the number of sample points. Instead, it is possible to increase the degree by splitting up the computation of the central sums for each sample point using iterative computation. For this, the set of traces of one sample point $\mathcal{Q}$ is partitioned into c subsets $\mathcal{Q}^{*i}$, $i \in \{1, \ldots, c\}$, the necessary central sums $CS_{d,\mathcal{Q}^{*i}}$ are computed for each subset in parallel incrementally, and all $CS_{d,\mathcal{Q}^{*i}}$ are combined using the iterative functions from Section 3.2.

4.5 Multivariate Evaluation

The equations presented in Section 4.4 only consider univariate settings. This is typically the case for hardware designs in which the shares are processed in parallel, and the sum of the leakages appear at a sample point. For software implementations this is usually not the case as the computations are sequential and split up over multiple clock cycles.

In this scenario the samples of multiple points in time are first combined using a combination function, and an attack is conducted on the combination's result. If the combination function (e.g., sum or product) does not require the mean, the extension of the equations to the multivariate case is trivial. It is enough to combine each set of samples separately and compute the mean and variance of the result iteratively as shown in the prior section.

However, this approach does not apply to the optimum combination function, i.e., the centered product [PRB09, SVO+10]. Given d sample point indices $\mathcal{J} = \{j_1, ..., j_d\}$ as points of interest and a set of sample vectors $\mathcal{Q} = \{\boldsymbol{V}_{i \in \{1,...,n\}}\}$ with $\boldsymbol{V}_i = \left(t_i^{(j)} \mid j \in \mathcal{J}\right)$, the centered product of the i-th trace is defined as

$$\prod_{j \in \mathcal{J}} \left(t_i^{(j)} - \mu_{\mathcal{Q}}^{(j)}\right), \tag{4.8}$$

where $\mu_{\mathcal{Q}}^{(j)}$ denotes the mean at sample point j over set $\mathcal{Q}$. The inclusion of the means is the reason why it is not easily possible to extend the equations from Section 4.4 to compute this value iteratively.

There is an iterative algorithm to compute the covariance similar to the aforementioned algorithms. This corresponds to the first parameter in a bivariate second-order scenario, i.e., $d = 2$. The covariance $\frac{SCP_{2,\mathcal{Q}'}}{n}$ is computed as shown in [Péb08] with

$$SCP_{2,\mathcal{Q}'} = SCP_{2,\mathcal{Q}} + \frac{n-1}{n}\left(y^{(1)} - \mu_{\mathcal{Q}}^{(1)}\right)\left(y^{(2)} - \mu_{\mathcal{Q}}^{(2)}\right) \tag{4.9}$$

for $\mathcal{Q}' = \mathcal{Q} \cup \{\left(y^{(1)}, y^{(2)}\right)\}$, $|\mathcal{Q}'| = n$, and an exemplary index set $\mathcal{J} = \{1, 2\}$. Still, even with this formula it is not possible to compute the required second parameter for the t-test. In the following, we present an extension of this approach to d sample points and show how this can be used to compute both parameters for a d-order d-variate t-test.

First, we define the sum of the centered products and the b-order power set of $\mathcal{J}$ which are required to compute the first parameter.

Definition 4.5.1. For d sample points and a set of sample vectors $\mathcal{Q}$, we denote the sum of the centered products as

$$SCP_{d,\mathcal{Q},\mathcal{J}'} = \sum_{\boldsymbol{V}_i \in \mathcal{Q}} \prod_{j \in \mathcal{J}'} \left(t_i^{(j)} - \mu_{\mathcal{Q}}^{(j)}\right). \tag{4.10}$$

Definition 4.5.2. The b-order power set of $\mathcal{J}$ is defined as

$$\mathcal{P}_k = \{\mathcal{S} \mid \mathcal{S} \in \mathbb{P}(\mathcal{J}), |\mathcal{S}| = k\}, \tag{4.11}$$

where $\mathbb{P}(\mathcal{J})$ refers to the power set of the indices of the points of interest $\mathcal{J}$.

Using these definitions we derive the following theorem.

Theorem 4.5.1. *Let $\mathcal{J}$ be a given set of indices (of d points of interest) and $\boldsymbol{V}$ the given sample vector with $\boldsymbol{V} = (y^{(1)}, ..., y^{(d)})$. The sum of the centered products $SCP_{d,\mathcal{Q}',\mathcal{J}}$ of the extended set $\mathcal{Q}' = \mathcal{Q} \cup \boldsymbol{V}$ with $\Delta^{(j \in \mathcal{J})} = y^{(j)} - \mu_{\mathcal{Q}}^{(j)}$ and $|\mathcal{Q}'| = n > 0$ can be computed as:*

$$\begin{aligned} SCP_{d,\mathcal{Q}',\mathcal{J}} = SCP_{d,\mathcal{Q},\mathcal{J}} &+ \left(\sum_{k=2}^{d-1} \sum_{\mathcal{S} \in \mathcal{P}_k} SCP_{k,\mathcal{Q},\mathcal{S}} \prod_{j \in \mathcal{J} \setminus \mathcal{S}} \left(\frac{\Delta^{(j)}}{-n}\right)\right) \\ &+ \left(\frac{(-1)^d(n-1) + (n-1)^d}{n^d} \prod_{j \in \mathcal{J}} \Delta^{(j)}\right). \end{aligned} \tag{4.12}$$

Proof. We start with the definition of the sum of the centered products and use $\mathcal{Q}' = \mathcal{Q} \cup \boldsymbol{V}$ to split up the term as

$$
\begin{aligned}
SCP_{d,\mathcal{Q}',\mathcal{J}} &= \sum_{\boldsymbol{V} \in \mathcal{Q}'} \prod_{j \in \mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}'}^{(j)} \right) \\
&= \left(\sum_{\boldsymbol{V} \in \mathcal{Q}} \prod_{j \in \mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}'}^{(j)} \right) \right) + \left(\prod_{j \in \mathcal{J}} \left(y^{(j)} - \mu_{\mathcal{Q}'}^{(j)} \right) \right). \qquad (4.13)
\end{aligned}
$$

Considering only the first term and using the relation $\mu_{\mathcal{Q}'}^{(j)} = \dfrac{(n-1)\,\mu_{\mathcal{Q}}^{(j)} + y^{(j)}}{n}$, we can write

$$
\begin{aligned}
\sum_{\boldsymbol{V} \in \mathcal{Q}} \prod_{j \in \mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}'}^{(j)} \right) &= \sum_{\boldsymbol{V} \in \mathcal{Q}} \prod_{j \in \mathcal{J}} \left(t^{(j)} - \frac{(n-1)\mu_{\mathcal{Q}}^{(j)} + y^{(j)}}{n} \right) \\
&= \sum_{\boldsymbol{V} \in \mathcal{Q}} \prod_{j \in \mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)} - \frac{\Delta^{(j)}}{n} \right) \\
&= \left(\sum_{\boldsymbol{V} \in \mathcal{Q}} \prod_{j \in \mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)} \right) \right) \\
&\quad + \left(\sum_{k=1}^{d-1} \sum_{\mathcal{S} \in \mathcal{P}_k} \sum_{\boldsymbol{V} \in \mathcal{Q}} \prod_{s \in \mathcal{S}} \left(t^{(s)} - \mu_{\mathcal{Q}}^{(s)} \right) \prod_{j \in \mathcal{J} \setminus \mathcal{S}} \frac{\Delta^{(j)}}{-n} \right) \\
&\quad + \left(\sum_{\boldsymbol{V} \in \mathcal{Q}} \prod_{j \in \mathcal{J}} \frac{\Delta^{(j)}}{-n} \right). \qquad (4.14)
\end{aligned}
$$

With Equation (4.10) and the fact that $\forall\ j \in \mathcal{J},\ \sum_{\boldsymbol{V} \in \mathcal{Q}} \left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)} \right) = 0$, we can simplify Equation (4.14) to

$$
\begin{aligned}
\sum_{\boldsymbol{V} \in \mathcal{Q}} \prod_{j \in \mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}'}^{(j)} \right) \; &= \; SCP_{d,\mathcal{Q},\mathcal{J}} + \left(\sum_{k=2}^{d-1} \sum_{\mathcal{S} \in \mathcal{P}_k} SCP_{k,\mathcal{Q},\mathcal{S}} \prod_{j \in \mathcal{J} \setminus \mathcal{S}} \frac{\Delta^{(j)}}{-n} \right) \\
&\quad + \; \frac{n-1}{(-n)^d} \prod_{j \in \mathcal{J}} \Delta^{(j)}. \qquad (4.15)
\end{aligned}
$$

The second term of Equation (4.13) can be reduced similarly as

$$
\begin{aligned}
\prod_{j \in \mathcal{J}} \left(y^{(j)} - \mu_{\mathcal{Q}'}^{(j)} \right) &= \prod_{j \in \mathcal{J}} \left(y^{(j)} - \frac{(n-1)\mu_{\mathcal{Q}}^{(j)} + y^{(j)}}{n} \right) \\
&= \prod_{j \in \mathcal{J}} \left(y^{(j)} - y^{(j)} + \frac{(n-1)\Delta^{(j)}}{n} \right) \\
&= \prod_{j \in \mathcal{J}} \left(\frac{(n-1)\Delta^{(j)}}{n} \right) \; = \; \frac{(n-1)^d}{n^d} \prod_{j \in \mathcal{J}} \Delta^{(j)}. \qquad (4.16)
\end{aligned}
$$

We can write Equation (4.13) by combining Equation (4.15) and Equation (4.16) as

$$\begin{aligned} SCP_{d,\mathcal{Q}',\mathcal{J}} &= SCP_{d,\mathcal{Q},\mathcal{J}} + \left(\sum_{k=2}^{d-1} \sum_{\mathcal{S}\in\mathcal{P}_k} SCP_{k,\mathcal{Q},\mathcal{S}} \prod_{J\in\mathcal{J}\setminus\mathcal{S}} \left(\frac{\Delta^{(j)}}{-n} \right) \right) \\ &+ \frac{n-1}{(-n)^d} \prod_{j\in\mathcal{J}} \Delta^{(j)} + \frac{(n-1)^d}{n^d} \prod_{j\in\mathcal{J}} \Delta^{(j)}, \end{aligned} \tag{4.17}$$

which is equivalent to Equation (4.12). □

Equation (4.12) can be also used to derive the second parameter of the t-tests. To this end, let us first recall the definition of the second parameter in the d-order d-variate case:

$$\begin{aligned} \sigma^2 &= \frac{1}{n} \sum_{\boldsymbol{V}\in\mathcal{Q}} \left(\prod_{j\in\mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)} \right) - \frac{SCP_{d,\mathcal{Q},\mathcal{J}}}{n} \right)^2 \\ &= \frac{1}{n} \left(\sum_{\boldsymbol{V}\in\mathcal{Q}} \prod_{j\in\mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)} \right)^2 \right) - \left(\frac{SCP_{d,\mathcal{Q},\mathcal{J}}}{n} \right)^2. \end{aligned} \tag{4.18}$$

The first term of the above equation can be written as

$$\begin{aligned} \frac{1}{n} \sum_{\boldsymbol{V}\in\mathcal{Q}} \prod_{j\in\mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)} \right)^2 &= \frac{1}{n} \sum_{\boldsymbol{V}\in\mathcal{Q}} \left(\prod_{j\in\mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)} \right) \prod_{j\in\mathcal{J}} \left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)} \right) \right) \\ &= \frac{SCP_{2d,\mathcal{Q},\mathcal{J}'}}{n}. \end{aligned} \tag{4.19}$$

Hence, the iterative algorithm (Equation (4.12)) can be performed with multiset $\mathcal{J}' = \{j_1, ..., j_d, j_1, ..., j_d\}$ to derive the first term of Equation (4.18). It is noteworthy that at the first glance Equation (4.18) looks like Equation (4.2), for which we addressed low accuracy issues. However, data which are processed by Equation (4.18) are already centered, that avoids the sums $SCP_{d,\mathcal{Q},\mathcal{J}}$ being very large values. Therefore, the accuracy issues which have been pointed out in Section 4.4 are evaded.

By combining the results of this section with that of Section 4.4, it is now possible to perform a t-test with any variate and at any order efficiently and with sufficient accuracy. As an example, we give all the formulas required by a second-order bivariate ($d = 2$) t-test in Appendix 12.6.

4.6 Case Studies

Security evaluations consist of the two phases *measurement* and *analysis*. All challenges regarding the second part, which in our scenario refers to the computation of the t-test statistics, have been discussed in detail in the previous sections. However, this alone does not ensure a correct evaluation as malpractice in the measurement phase can lead to faulty results in the analysis. Below, we first describe the pitfalls that can occur during the measurement phase and provide solutions to ensure the correctness of evaluations. After that, two case studies are discussed that exemplary show the applications of our proposed evaluation framework.

4.6.1 Framework

If the DUT is equipped with countermeasures, the evaluation might require the collection of many (millions of) traces and, thus, the measurement rate (i.e., the number of collected traces per a certain period) can become a major hurdle. Following the technique suggested in [CDG$^+$13, KW13] we explain how the measurement phase can be significantly accelerated. The general scenario (cf. Figure 4.2) is based on the assumption that the employed acquisition device (e.g., oscilloscope) includes a feature usually called *sequence mode* or *rapid block mode.* In such a mode – depending on the length of each trace as well as the size of the sampling memory of the oscilloscope – the acquisition device can record multiple traces. This is beneficial since the biggest bottleneck in the measurement phase is the low speed of the communication between e.g., the PC and the DUT (usually realized by UART). In the scenario shown in Figure 4.2 it is supposed that **Target** is the DUT, and **Control** a microcontroller (or an FPGA) which communicates with the DUT as well as with the PC. The terms **Target** and **Control** correspond to the two FPGAs of e.g., a SAKURA (SASEBO) platform [Sak], but in some frameworks these two parties are merged, e.g., a microcontroller-based platform. Further, the PC is already included in modern oscilloscopes.

Profiting from the sequence mode the communication between the PC and the DUT can be minimized in such a way that the PC sends only a single request to collect multiple N traces. The measurement rate depends on the size of the oscilloscope's sampling memory, the length of each trace as well as the frequency of operation of the DUT. As an example, by means of an oscilloscope with 64 MByte sampling memory (per channel) we are able to measure $N = 10,000$ traces per request when each trace consists of $5,000$ sample points. This results in being able to collect 100 million traces (for either a specific or non-specific t-test) in 12 hours. We should point out that the given scenario is not specific to t-test evaluations. It can also be used to speed up the measurement process in case of an evaluation by state-of-the-art attacks when the key is known.

To assure the correctness of the measurements, the PC should be able to follow and verify the processes performed by both **Control** and the DUT. Our suggestion is to employ a random number generator which can be seeded by the PC[1]. This allows the PC to check the consistency of out_N as well as the Pseudo-Random Number Generator (PRNG) state. With respect to Figure 4.2, $f(.,.,.)$ is defined based on the desired evaluation scheme. For a specific t-test (or any evaluation method where no control over e.g., plaintexts is necessary) our suggestion is:

$$in_{i+1} \;=\; f(\texttt{INPUT},\; out_i,\; random) \;=\; out_i \oplus random.$$

This allows the PC to verify all N processes of the DUT by examining the correctness of out_N. In case of a non-specific t-test, such a function can be realized as

$$in_{i+1} \;=\; f(\texttt{INPUT},\; out_i,\; random) \;=\begin{cases} \texttt{INPUT} & \text{if } random_{bit} \text{ is } 0 \\ random & \text{if } random_{bit} \text{ is } 1 \end{cases}.$$

Note that it should be ensured that $random_{bit}$ is excluded from the random input. Otherwise, the random inputs become biased at a certain bit which may potentially lead to false-positive evaluation results. If a semi-fixed vs. random t-test is conducted, `INPUT` contains a set of certain

---

[1] For example an AES encryption engine in counter mode.

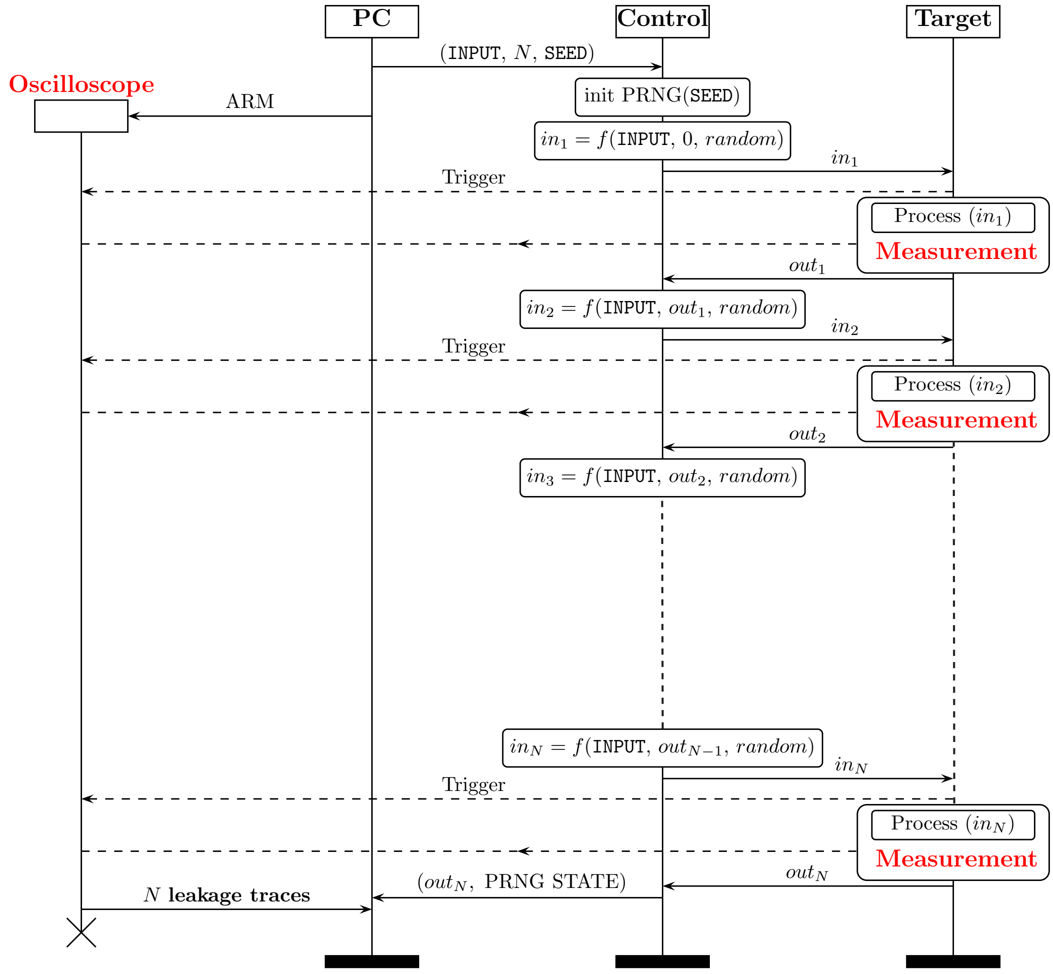


Figure 4.2: An optimized measurement process.

fixed inputs (which can be stored in **Control** to reduce the communications), and the function can be implemented as

$$in_{i+1} \;=\; f(\mathtt{INPUT},\ out_i,\ random) \;=\; \begin{cases} \mathtt{INPUT}_{random} & \text{if } random_{bit} \text{ is } 0 \\ random & \text{if } random_{bit} \text{ is } 1 \end{cases} .$$

If the DUT is equipped with masking countermeasures, all communication between **Control** and **Target** (and preferably with the PC as well) should be in a shared form. This prevents the unshared data, e.g., `INPUT`, from appearing in **Control** and **Target**. Otherwise, the leakage associated to the input itself would cause, e.g., a non-specific t-test to report an exploitable leakage regardless of the robustness of the DUT. In hardware platforms such a shared communication is essential due to the static leakage as well [Mor14a]. For instance, in a second-order masking scheme (where variables are represented by three shares) `INPUT` should be a 3-share value $(\texttt{INPUT}^1, \texttt{INPUT}^2, \texttt{INPUT}^3)$, and respectively $in_{i+1} = (in^1_{i+1}, in^2_{i+1}, in^3_{i+1})$. In such a case, a non-specific t-test (including semi-fixed vs. random) should be handled as

$$\begin{aligned} in_{i+1} &= f(\texttt{INPUT},\ out_i,\ random) \\ &= \begin{cases} (\texttt{INPUT}^1 \oplus r^1, \texttt{INPUT}^2 \oplus r^2, \texttt{INPUT}^3 \oplus r^1 \oplus r^2) & \text{if } random_{bit} \text{ is } 0 \\ (r^1, r^2, r^3) & \text{if } random_{bit} \text{ is } 1 \end{cases}, \end{aligned}$$

with r^1 as a short notation of $random^1$. In other words, the fixed input should be freshly remasked before being sent to the DUT. Consequently, the last output $(out^1_N, out^2_N, out^3_N)$ is also sent in a shared form to the PC.

In general, we suggest applying the tests with the following settings:

- non-specific t-test (fixed vs. random): with shared communication between the parties, if the DUT is equipped with masking.
- non-specific t-test (semi-fixed vs. random): without shared communication, if the DUT is equipped with hiding techniques.
- specific t-tests: with the goal of identifying a suitable intermediate value for a key-recovery attack, if the DUT is not equipped with any countermeasures or failed in former non-specific tests. In this case, a shared communication is preferable if the DUT is equipped with masking.

4.6.2 Case Study: Microcontroller

As the first case study we consider the publicly-available implementation of the DPA contest v4.2 [TEL15] for an Atmel microcontroller. The underlying implementation is a realization of the AES-128 encryption engine equipped with masking and shuffling. The details of the implementation can be found in [BBD$^+$14]; we also give the pseudo-code in Algorithm 1. It is noteworthy that the underlying countermeasure is based on a low-entropy masking scheme [NSGD12] which uses 8-bit masks drawn from a 16-element set. Further, the shuffling (of the order of the masked Sbox look-ups) is only applied to the first and last rounds. Indeed, the implementation is a revised version of the DPA contest v4.1 after the flaws reported in [MGH14].

By means of a PicoScope at the sampling rate of 250 MS/s we collected 100, 000 traces of this implementation running on an ATmega163-based smartcard. The traces have been measured using the aforementioned framework for a non-specific t-test, and each trace covers only the first two encryption rounds. Note that since the underlying implementation receives unmasked plaintext and performs the masking (on the key) prior to the encryption (see Algorithm 1), we were not able to completely follow the instructions (communication in a shared form) suggested above. In other to follow a shared communication fashion, one needs to slightly modify the implementation. By performing the first- and second-order univariate non-specific t-tests we obtained the results shown in Figure 4.3. As expected, the leakage associated to the unmasked plaintext before being XORed with the masked roundkey can be identified in the t-test result, i.e., the time period between 0 and 20 μs. The test also shows that the implementation still exhibits first-order leakage even in the first round, where both masking and shuffling are active. As expected, when the process is not shuffled (i.e., the second encryption round), the leakage is

Algorithm 1: Masked and Shuffled AES-128 encryption.

```
Input  : Plaintext $X$, seen as bytes $x_{i \in \{0,\dots,15\}}$,
         11 Roundkeys $R[r]$, $r \in \{0,\dots,10\}$, each 128-bit constant
Output: Ciphertext $X$, seen as bytes $x_{i \in \{0,\dots,15\}}$

Draw 16 random $\texttt{offset}_{i \in \{0,\dots,15\}}$ uniformly in $\{0,1\}^4$
Draw two random bijective table $\texttt{Shuffle0}, \texttt{Shuffle10} : \{0,1\}^4 \rightarrow \{0,1\}^4$

$R[0] = R[0] \oplus \texttt{Mask}[\texttt{offset}]$
for $r \in \{0, 10\}$ do
    $X = X \oplus R[r]$
    for $i \in \{0, 15\}$ do
        if $r = 0$ then $j = \texttt{Shuffle0}[i]$
        else if $r = 10$ then $j = \texttt{Shuffle10}[i]$
        else $j = i$
        $x_j = \texttt{MaskedSbox}_{\ \texttt{offset}_j}(x_j)$
    end
    if $r \neq 10$ then
        $X = \texttt{MixColumn}(\texttt{ShiftRows}(X))$
        $X = X \oplus \texttt{MaskCompensation}(\texttt{offset})$
    else
        $X = \texttt{ShiftRows}(X)$
        $X = X \oplus \texttt{MaskCompensationLastRound}(\texttt{offset})$
    end
end
```

detectable with higher confidence. Since our goal is just to assess the leakage of the implementation, we have not tried to identify a suitable intermediate value nor a hypothetical (power) model for a successful key-recovery attack.

4.6.3 Case Study: FPGA

For the second case study we focus on a second-order TI as described in [Rep15]. The initial higher-order TI constructions as given in [BGN$^+$14b] considers only univariate leakage and the note by Reparaz addresses this issue that multivariate leakages can still be exploited from a higher-order TI design. In order to examine this by a multivariate t-test (explained in Section 4.5) we implemented the Non-Linear Feedback Shift Register (NLFSR) which has been taken as an example in [Rep15]. The NLFSR consists of four cells $L[0]$ to $L[3]$ and an AND/XOR module as the feedback function

$$f = f(L[3], L[2], L[1]) = L[3] \oplus L[2]\, L[1],$$

which feeds the $L[0]$ cell. We followed the concept of second-order TI and took the uniform sharing of the AND/XOR module from [BGN$^+$14b], which needs at least 5 input shares. We implemented the design (cf. Figure 4.4) on a SAKURA-G [Sak] platform with a Spartan-6

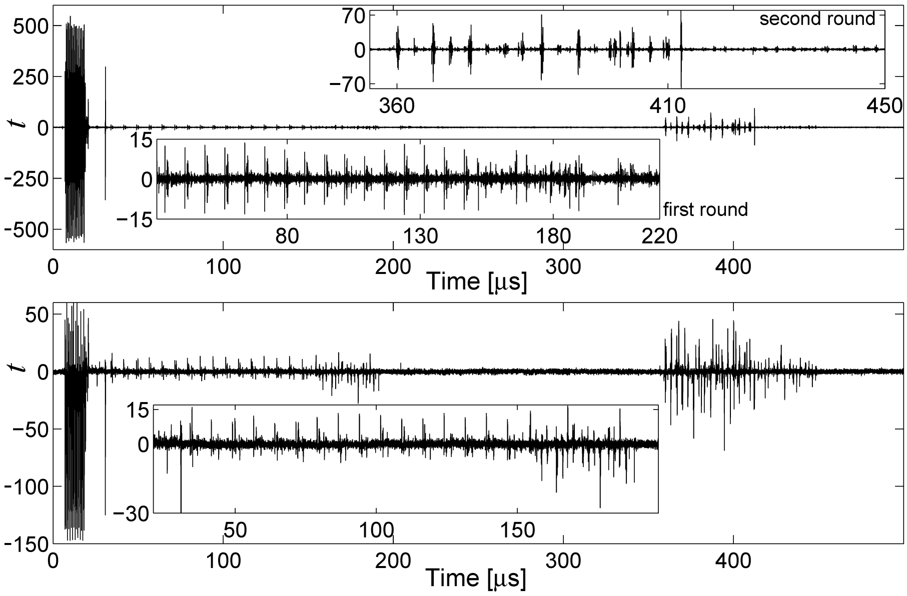


Figure 4.3: DPA contest v4.2, non-specific t-test results (top) first-order, (bottom) second-order univariate using 100, 000 traces.

FPGA as the target (DUT). The NLFSR is initialized by a 4-bit input each represented by 5 shares, and it is clocked 32 times till the 4-bit (shared) output is generated.

In order to conduct a non-specific t-test we followed the measurement scenario presented in Section 4.6.1, where all the communications are shared. In total, we collected 2, 000, 000 power traces (an exemplary one is shown by Figure 4.5(a)) at a sampling rate of 500 MS/s. By performing the univariate fixed vs. random t-test at first up to fifth orders we obtained the curves of the statistics which are shown in Figure 4.5(b) to Figure 4.5(f). As expected, the design exhibits a fifth-order leakage as the underlying masking utilizes 5 shares. For a bivariate second-order t-test we followed the method explained in Section 4.5 with $d = 2$ (the formulas to derive both parameters of a bivariate second-order t-test are given in Appendix 12.6). Since

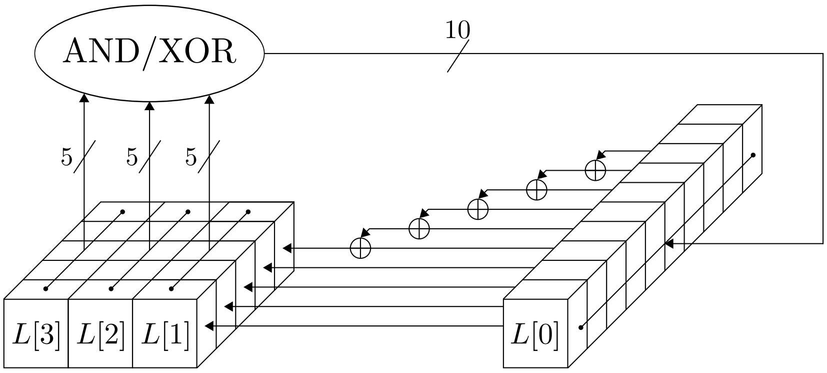
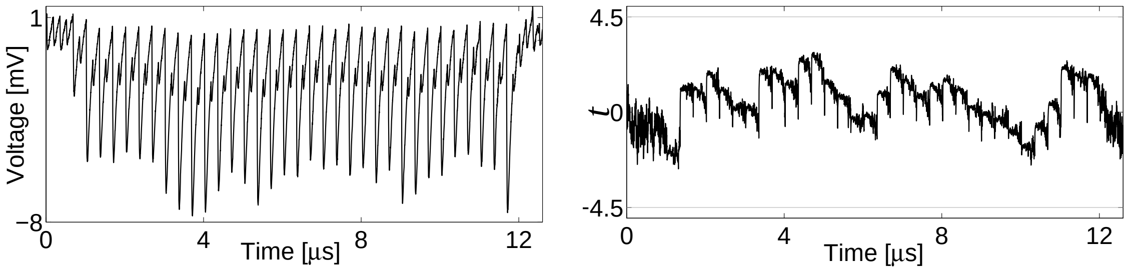


Figure 4.4: Architecture of the second-order TI of the NLFSR.

the selection of the points of interest in a bivariate setting is not trivial (one can also use the scheme introduced in [RGV12] or in [DSV$^+$15]), we have examined all possible offsets (between two points of interest) from 1 up to 31 clock cycles, and performed the test on all sample points of the collected traces. The test with respect to 15 clock cycles as the offset between the points of interest showed the best result as depicted in Figure 4.5(g). With this experiment we could practically confirm the issue of higher-order TI addressed in [Rep15] with a bivariate second-order non-specific t-test (without performing any key-recovery attack).

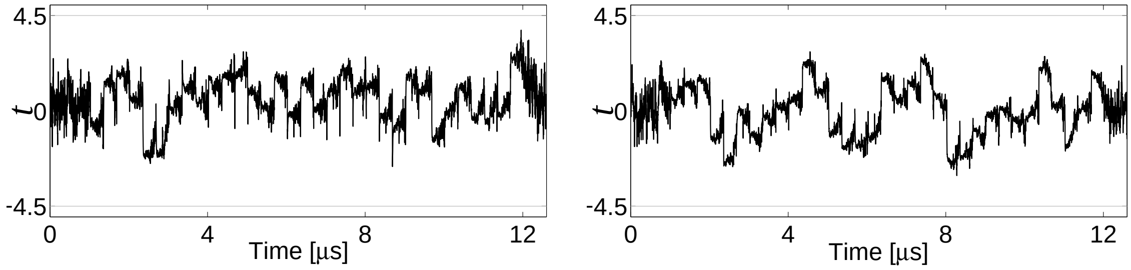
4.7 Conclusion

Security evaluations using Welch's t-test have become popular in recent years. In this chapter, we have extended the theoretical foundations and guidelines regarding the leakage assessment introduced in [GJJR11]. In particular, we have given more detailed instructions how the test can be applied in a higher-order setting by highlighting problems that can occur during the computation of this test. Additionally, we have discussed and provided guidelines for an optimized measurement setup which allows high measurement rate and avoids faulty evaluations.



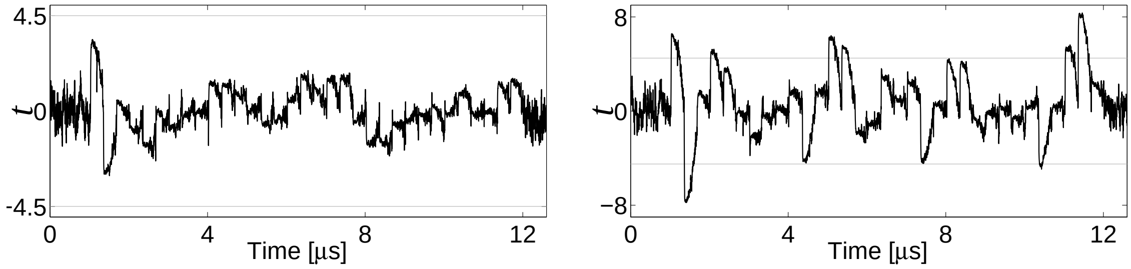
(a) Sample Trace

(b) first-order



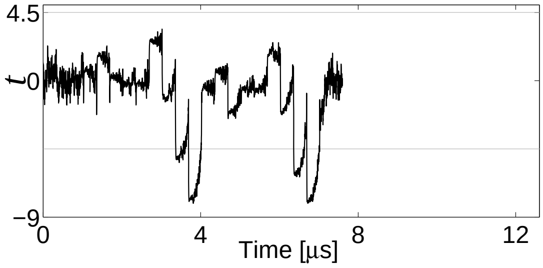
(c) second-order

(d) third-order



(e) fourth-order

(f) fifth-order



(g) bivariate second-order, 15 clock cycles offset

Figure 4.5: NLFSR 2nd-order TI, sample trace and univariate and bivariate non-specific t-test results using $2,000,000$ traces.

Chapter 5

Robust and One-Pass Parallel Computation of Correlation-based Attacks

In this chapter, we present our contributions related to attack-based evaluation methodologies for side-channel countermeasures based on [SMG16c]. We introduce procedures that allow to iteratively compute Pearson's correlation coefficient in the scenario of a side-channel analysis at arbitrary orders. The advantages of our proposed solutions are the same as for the computation methodology of Welch's t-test in Chapter 4. In short, our constructions allow efficiently performing higher-order side-channel analysis attacks (e.g., on hundreds of million traces) which is of crucial importance when practical evaluation of the masking schemes need to be performed.

Contents of this Chapter

5.1 Introduction **45**
5.2 Notations **46**
5.3 Univariate CPA **47**
5.4 Multivariate CPA **51**
5.5 Moments-Correlating DPA **54**
5.6 Evaluation **59**
5.7 Conclusion **61**

5.1 Introduction

The non-specific test-based evaluation approach presented in the previous chapter can only report the *existence* of leakage in a product, but it does not necessarily provide any indication whether this leakage is *exploitable* by an actual attack. In reply to the question if a leakage is in fact exploitable for key recovery, it is required to mount different SCA attacks and examine their success. Depending on the definition and settings of the masking scheme, it can provide security against SCA attacks up to a certain order d. Consequently, all tests and attacks need to take all particular orders ranging from 1 up to $d+1$ into account.

The most common SCA attack (CPA [BCO04]) is based on a hypothetical leakage model and the estimation of the correlation (commonly by Pearson's correlation coefficient) between the hypothetical leakages and the SCA traces. In its simplest setting, the attack runs independently at each sample point of the SCA traces. This univariate first-order CPA can be extended to higher orders $d > 1$ by introducing a preprocessing stage for the traces.

As shown in [BB16] for first-order and second-order CPA, the formulas for preprocessing and the estimation of the correlation can be combined by following the displacement law. Their iterative computation procedure is based on the raw sums solves all the shortcomings of the three-pass approach. In fact:

- When increasing the trace pool, the estimated raw moments are easily updated by only processing the given new traces.
- The attack can be started before the measurement phase is completed. This helps to further increase the performance of the attacks.
- The result of the attack can be obtained without introducing any overhead to the process of the further traces at any time during the measurement phase.
- The trace pool can be easily split into smaller sets and each set can be processed independently by different threads. Due to the nature of the raw moments, the result of different threads (at any time) can be easily combined to derive the result of the attack.

Note, however, that this procedure was only presented for first-order and bivariate second-order CPA using 10,000,000 traces and may suffer from the aforementioned numerical instabilities as the raw moments become pretty large values by increasing the number of traces.

5.1.1 Contribution

In this work, we present an approach based on central and standardized moments to cover univariate as well as multivariate CPA attacks at any arbitrary order. Our solution benefits from all the aforementioned advantages of the raw-moment approach while it maintains the accuracy (as for the three-pass approach) regardless of the order of the attack and the number of traces. This work not only covers CPA attacks but also Moments-Correlating DPA (MC-DPA) [MS14] where moments are correlated to the (preprocessed) traces with the goal of avoiding the necessity of a hypothetical leakage model (that is unavoidable in CPA attacks).

5.2 Notations

We use capital letters for random variables, and lower-case letters for their realizations. Vectors are denoted with bold notations, functions with sans serif fonts, and sets with calligraphic ones.

Suppose that in a side-channel attack, with respect to n queries with associated data (e.g., plaintext or ciphertext) $\boldsymbol{d}_{i\in\{1,\ldots,n\}}$, n side-channel measurements (so-called traces) are collected. Let us denote each trace by $\boldsymbol{t}_{i\in\{1,\ldots,n\}}$ containing m sample points $\{t_i^{(1)}, \ldots, t_i^{(m)}\}$.

Following the divide-and-conquer principle, one objective of a side-channel attack is to recover a part k of the secret key $\boldsymbol{k}$, which contributed to the processing of the entire associated data $\boldsymbol{d}_{i\in\{1,\ldots,n\}}$. Prior to the attack an intermediate value V is selected, which given the associated data and a key guess k is predictable, i.e., $v_i = \mathsf{F}(\boldsymbol{d}_i, k)$. In a CPA attack a hypothetical leakage model $\widetilde{\mathsf{L}}(.)$ is applied on the chosen intermediate value which should be (sufficiently) linearly proportional to the actual leakage of the target device, i.e., $\mathsf{L}(.)$. As a common and straightforward example, the Hamming weight of an Sbox output during the first round of an encryption function is employed when attacking an exemplary micro-processor based implementation, i.e., $l_i = \widetilde{\mathsf{L}}(v_i) = HW\left(\mathsf{S}\left(d_i \oplus k\right)\right)$, where d_i denotes a necessary part of $\boldsymbol{d}_i$ to predict v_i.

5.3 Univariate CPA

For a *univariate* CPA attack the correlation between the traces $\boldsymbol{T}$ and the hypothetical leakage values L is estimated. Due to the *univariate* nature of the attack, such a process is performed at each sample point $(1, \dots, m)$ independently. Therefore, below – for simplicity – we omit the upper index of the sample points and denote a sample point of the ith trace by t_i.

The estimation of the correlation with Pearson's correlation coefficient (as the normalized covariance) is defined as

$$\rho = \frac{\operatorname{cov}(T, L)}{\sigma_t \, \sigma_l} = \frac{\mathsf{E}\big(\,(T - \mu_t)\,(L - \mu_l)\,\big)}{\sigma_t \, \sigma_l}, \tag{5.1}$$

where μ_t (resp. μ_l) denotes the estimated mean of the traces (resp. of the hypothetical leakages). σ_t (resp. σ_l) also stands for standard deviation.

In the discrete domain we can write

$$\rho = \frac{\frac{1}{n} \sum_{i=1}^{n} (t_i - \mu_t)(l_i - \mu_l)}{\sqrt{\frac{1}{n} \sum_{i=1}^{n} (t_i - \mu_t)^2 \, \frac{1}{n} \sum_{i=1}^{n} (l_i - \mu_l)^2}} \tag{5.2}$$

Based on the way followed in [BB16] one can write

$$\rho = \frac{\frac{1}{n} \sum_{i=1}^{n} t_i \, l_i - \mu_t \, \mu_l}{\sqrt{\left(\frac{1}{n} \sum_{i=1}^{n} {t_i}^2 - {\mu_t}^2\right) \left(\frac{1}{n} \sum_{i=1}^{n} {l_i}^2 - {\mu_l}^2\right)}} = \frac{M_{1,\mathcal{T}\cdot\mathcal{L}} - M_{1,\mathcal{T}} \, M_{1,\mathcal{L}}}{\sqrt{\left(M_{2,\mathcal{T}} - {M_{1,\mathcal{T}}}^2\right) \left(M_{2,\mathcal{L}} - {M_{1,\mathcal{L}}}^2\right)}}, \tag{5.3}$$

which are based on d-order raw moments, i.e., $M_{d,\mathcal{X}} = \frac{1}{n} \sum_{i=1}^{n} {x_i}^d$. However, as shown in Chapter 4, such constructions can lead to numerically unstable situations [Hig02]. During the computation of the raw moments the intermediate values tend to become very large which can lead to a loss in accuracy. Further, M_2 and ${M_1}^2$ can be large values, and the result of $M_2 - {M_1}^2$ can also lead to a significant accuracy loss due to the limited fraction significand of floating point formats (e.g., IEEE 754).

Iterative.

We can alternatively write

$$\rho = \frac{\frac{1}{n} \sum_{i=1}^{n} (t_i - \mu_t)(l_i - \mu_l)}{\sqrt{\frac{1}{n} \sum_{i=1}^{n} (t_i - \mu_t)^2 \, \frac{1}{n} \sum_{i=1}^{n} (l_i - \mu_l)^2}} = \frac{\frac{1}{n} ACS_1}{\sqrt{\frac{1}{n} CS_{2,\mathcal{T}} \, \frac{1}{n} CS_{2,\mathcal{L}}}}, \tag{5.4}$$

with ACS_1 as the first-order *adjusted central sum.*

Suppose that $\mathcal{Q}_1$ and $\mathcal{Q}_2$ denote sets of doubles (t, l) with first-order adjusted central sum $ACS_{1,\mathcal{Q}_1}$ and $ACS_{1,\mathcal{Q}_2}$ respectively. The first-order adjusted central sum of $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$ can be written as

$$ACS_{1,\mathcal{Q}} = ACS_{1,\mathcal{Q}_1} + ACS_{1,\mathcal{Q}_2} + \frac{n_1\, n_2}{n}\,\Delta_t\,\Delta_l, \tag{5.5}$$

with $\Delta_t = \mu_{t,\mathcal{Q}_2} - \mu_{t,\mathcal{Q}_1}$ and $\Delta_l = \mu_{l,\mathcal{Q}_2} - \mu_{l,\mathcal{Q}_1}$. For simplicity, we denote $M_{1,\mathcal{T}_1}$ by $\mu_{t,\mathcal{Q}_1}$ and $M_{1,\mathcal{L}_1}$ by $\mu_{l,\mathcal{Q}_1}$. The sets $\mathcal{T}_1$ and $\mathcal{L}_1$ are formed respectively from the first and second elements of the doubles in $\mathcal{Q}_1$ (the same holds for $\mathcal{Q}_2$, $\mu_{t,\mathcal{Q}_2}$, and $\mu_{l,\mathcal{Q}_2}$).

With the above given formulas the estimation of the correlation in a first-order CPA attack can be efficiently parallelized. The traces can be split into small sets, and with the mean, second-order central sum, and first-order adjusted central sum of each set, the final correlation can be easily estimated.

Incremental, $n_2 = 1$.

We now optimize the computations of each set. It is indeed enough to suppose that $\mathcal{Q}_2$ consists of only one element. The incremental functions for the raw moments and central sums were already discussed in the previous chapter. For the first-order adjusted central sum we can write

$$ACS_{1,\mathcal{Q}} = ACS_{1,\mathcal{Q}_1} + \frac{n-1}{n}\,\Delta_t\,\Delta_l, \tag{5.6}$$

with $\Delta_t = t_n - \mu_{t,\mathcal{Q}_1}$ and $\Delta_l = l_n - \mu_{l,\mathcal{Q}_1}$, where $\mathcal{Q}_2 = \{(t_n, l_n)\}$.

Based on these formulas the correlation can be computed efficiently in one pass. Furthermore, since the intermediate results of the central sums are mean-free, they do not become significantly large which helps to prevent the numerical instabilities.

5.3.1 Univariate Higher-Order CPA

Higher-order attacks require that the sample traces are preprocessed. For the second-order univariate CPA the preprocessing consists of making each sample point mean-free squared:

$$t'_i = (t_i - \mu_t)^2 .$$

For higher orders $d > 2$ the traces are usually additionally standardized as $\frac{t'_i}{\sigma_t{}^d}$, where σ_t denotes the standard deviation. Therefore, the Pearson's correlation coefficient can be written as

$$\rho = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{t'_i}{\sigma_t{}^d} - \frac{\mu_{t'}}{\sigma_t{}^d}\right)(l_i - \mu_l)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{t'_i}{\sigma_t{}^d} - \frac{\mu_{t'}}{\sigma_t{}^d}\right)^2 \frac{1}{n}\sum_{i=1}^{n}(l_i - \mu_l)^2}} = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(t'_i(l_i - \mu_l)\right)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(t'_i - \mu_{t'})^2 \frac{1}{n}\sum_{i=1}^{n}(l_i - \mu_l)^2}} \tag{5.7}$$

The straightforward way is to first preprocess the entire trace set $t_{i\in\{1,\ldots,n\}}$. Hence, the measurement phase has to be completed before the preprocessing can be started. Another drawback is the reduced efficiency as each of the preprocessing and the estimation of the correlation steps needs at least one pass over the whole trace set.

In [BB16], the authors propose iterative formulas for first- and second-order CPA. Their approach is based on raw moments which can lead to numerical instability if the values get too large [SM16]. Alternatively, we propose an iterative method which is based on the central moments. These values are mean-free which leads to smaller values and better accuracy for a large number of measurements. This approach can be run in parallel to the measurements (and can be also split into smaller threads) as the result is incrementally updated for each new measurement. Therefore, it needs only one pass over the whole trace set. In the following, we present all necessary iterative formulas to perform a univariate CPA at any arbitrary order with sufficient accuracy. We divide the expressions by the numerator and denominator of Equation (5.7).

5.3.2 Numerator

Note that even though the numerator looks similar to a raw-moment approach, it operates with central (mean-free) values. Therefore, numerical instabilities are avoided. The numerator for the d-th order correlation can be written as

$$\frac{1}{n}\sum_{i=1}^{n}\left(t'_i(l_i-\mu_l)\right)=\frac{1}{n}\sum_{i=1}^{n}\left(t_i-\mu_t\right)^d\left(l_i-\mu_l\right)=\frac{1}{n}ACS_d, \tag{5.8}$$

with ACS_d which we refer to as the d-order *adjusted central sum.*

We start with a generic formula which merges the adjusted central sum of two sets $\mathcal{Q}_1\cup\mathcal{Q}_2=\mathcal{Q}$ with $|\mathcal{Q}_1|=n_1$, $|\mathcal{Q}_2|=n_2$ and $|\mathcal{Q}|=n$. The goal is to compute $ACS_{d,\mathcal{Q}}$ given only the adjusted and central sums of $\mathcal{Q}_1$ and $\mathcal{Q}_2$.

Theorem 5.3.1. *Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be given sets of doubles (t,l). Suppose also $\mathcal{T}_1$ and $\mathcal{L}_1$ as the sets of respectively the first and second elements of the doubles in $\mathcal{Q}_1$ (the same for $\mathcal{T}_2$ and $\mathcal{L}_2$). The d-order adjusted central sum $ACS_{d,\mathcal{Q}}$ of the extended set $\mathcal{Q}=\mathcal{Q}_1\cup\mathcal{Q}_2$ with $\Delta_t=\mu_{t,\mathcal{Q}_2}-\mu_{t,\mathcal{Q}_1}$ and $\Delta_l=\mu_{l,\mathcal{Q}_2}-\mu_{l,\mathcal{Q}_1}$ can be written as*

$$\begin{aligned}ACS_{d,\mathcal{Q}}=&\ ACS_{d,\mathcal{Q}_1}+ACS_{d,\mathcal{Q}_2}+\frac{\Delta_l}{n}\left(n_1\,CS_{d,\mathcal{Q}_2}-n_2\,CS_{d,\mathcal{Q}_1}\right)\\&+\sum_{p=1}^{d-1}\binom{d}{p}\left(\frac{\Delta_t}{n}\right)^p\Big[(-n_2)^p\,ACS_{d-p,\mathcal{Q}_1}+(n_1)^p\,ACS_{d-p,\mathcal{Q}_2}\\&+\frac{\Delta_l}{n}\Big((-n_2)^{p+1}\,CS_{d-p,\mathcal{Q}_1}+(n_1)^{p+1}\,CS_{d-p,\mathcal{Q}_2}\Big)\Big]\\&+\frac{\left(n_1\,(-n_2)^{d+1}+n_2\,(n_1)^{d+1}\right)}{n^{d+1}}\left(\Delta_t\right)^d\,\Delta_l\end{aligned} \tag{5.9}$$

Proof. We start with the definition of $ACS_{d,\mathcal{Q}}$ based on Equation (5.8) and write

$$\begin{aligned}ACS_{d,\mathcal{Q}}&=\sum_{(t_i,l_i)\in\mathcal{Q}}\left(t_i-\mu_{t,\mathcal{Q}}\right)^d\left(l_i-\mu_{l,\mathcal{Q}}\right)\\&=\sum_{(t_i,l_i)\in\mathcal{Q}_1}\left(t_i-\mu_{t,\mathcal{Q}}\right)^d\left(l_i-\mu_{l,\mathcal{Q}}\right)+\sum_{(t_i,l_i)\in\mathcal{Q}_2}\left(t_i-\mu_{t,\mathcal{Q}}\right)^d\left(l_i-\mu_{l,\mathcal{Q}}\right).\end{aligned} \tag{5.10}$$

We first find iterative formulas for each sum separately and combine them in the end. Starting with the first sum of Equation (5.10) we use the definition of $M_{1,\mathcal{Q}}$ given in Equation (3.8) for $\mu_{t,\mathcal{Q}}$ as well as $\mu_{l,\mathcal{Q}}$ and write

$$\begin{aligned}
&\sum_{(t_i,l_i)\in\mathcal{Q}_1} (t_i-\mu_{t,\mathcal{Q}})^d\,(l_i-\mu_{l,\mathcal{Q}}) = \\
&\sum_{(t_i,l_i)\in\mathcal{Q}_1} \left(t_i-\frac{n_1\,\mu_{t,\mathcal{Q}_1}+n_2\,\mu_{t,\mathcal{Q}_2}}{n}\right)^d \left(l_i-\frac{n_1\,\mu_{l,\mathcal{Q}_1}+n_2\,\mu_{l,\mathcal{Q}_2}}{n}\right) = \\
&\sum_{(t_i,l_i)\in\mathcal{Q}_1} \left(t_i-\mu_{t,\mathcal{Q}_1}-\frac{n_2}{n}\Delta_t\right)^d \left(l_i-\mu_{l,\mathcal{Q}_1}-\frac{n_2}{n}\Delta_l\right)
\end{aligned} \tag{5.11}$$

Following [Péb08] we write the first term of the product of Equation (5.11) as

$$\begin{aligned}
\left(t_i-\mu_{t,\mathcal{Q}_1}-\frac{n_2}{n}\Delta_t\right)^d &= (t_i-\mu_{t,\mathcal{Q}_1})^d + \sum_{p=1}^{d-1}\binom{d}{p}(t_i-\mu_{t,\mathcal{Q}_1})^{d-p}\left(-\frac{n_2}{n}\Delta_t\right)^p \\
&\quad + \left(-\frac{n_2}{n}\Delta_t\right)^d
\end{aligned} \tag{5.12}$$

By combining Equation (5.11) and Equation (5.12) we derive

$$\begin{aligned}
&\sum_{(t_i,l_i)\in\mathcal{Q}_1} (t_i-\mu_{t,\mathcal{Q}})^d\,(l_i-\mu_{l,\mathcal{Q}}) = \\
&ACS_{d,\mathcal{Q}_1} + \sum_{p=1}^{d-1}\binom{d}{p} ACS_{d-p,\mathcal{Q}_1}\left(-\frac{n_2}{n}\Delta_t\right)^p + CS_{d,\mathcal{Q}_1}\left(-\frac{n_2}{n}\Delta_l\right) \\
&+\sum_{p=1}^{d-2}\binom{d}{p} CS_{d-p,\mathcal{Q}_1}\left(-\frac{n_2}{n}\Delta_t\right)^p\left(-\frac{n_2}{n}\Delta_l\right) + n_1\left(-\frac{n_2}{n}\Delta_t\right)^d\left(-\frac{n_2}{n}\Delta_l\right)
\end{aligned} \tag{5.13}$$

This can be simplified to

$$\begin{aligned}
&ACS_{d,\mathcal{Q}_1} + CS_{d,\mathcal{Q}_1}\left(-\frac{n_2}{n}\Delta_l\right) \\
&+\sum_{p=1}^{d-1}\binom{d}{p}\left(-\frac{n_2}{n}\Delta_t\right)^p\left[ACS_{d-p,\mathcal{Q}_1} + CS_{d-p,\mathcal{Q}_1}\left(-\frac{n_2}{n}\Delta_l\right)\right] \\
&+ n_1\left(-\frac{n_2}{n}\right)^{d+1}(\Delta_t)^d\,\Delta_l
\end{aligned} \tag{5.14}$$

It is noteworthy that $CS_{1,\mathcal{Q}_1}$ is always zero and is ignored in the above expression.

This procedure is repeated for the second sum of Equation (5.10), and we derive

$$\begin{aligned}
\sum_{(t_i,l_i)\in\mathcal{Q}_2} (t_i-\mu_{t,\mathcal{Q}})^d\,(l_i-\mu_{l,\mathcal{Q}}) &= ACS_{d,\mathcal{Q}_2} + CS_{d,\mathcal{Q}_2}\left(\frac{n_1}{n}\Delta_l\right) \\
&\quad+\sum_{p=1}^{d-1}\binom{d}{p}\left(\frac{n_1}{n}\Delta_t\right)^p\left[ACS_{d-p,\mathcal{Q}_2} + CS_{d-p,\mathcal{Q}_2}\left(\frac{n_1}{n}\Delta_l\right)\right] \\
&\quad+ n_2\left(\frac{n_1}{n}\right)^{d+1}(\Delta_t)^d\,\Delta_l
\end{aligned} \tag{5.15}$$

By combining Equation (5.14) and Equation (5.15) we obtain Equation (5.9).

□

Incremental, $n_2 = 1$.

For the iterative formulas when $\mathcal{Q}_2 = \{(t_n, l_n)\}$ Equation (5.9) can be simplified to

$$\begin{aligned}ACS_{d,\mathcal{Q}} =& ACS_{d,\mathcal{Q}_1} + CS_{d,\mathcal{Q}_1}\left(-\frac{\Delta_l}{n}\right)\\ &+\sum_{p=1}^{d-1}\binom{d}{p}\left(-\frac{\Delta_t}{n}\right)^p\left[ACS_{d-p,\mathcal{Q}_1} + CS_{d-p,\mathcal{Q}_1}\left(-\frac{\Delta_l}{n}\right)\right]\\ &+\frac{(-1)^{d+1}(n-1)+(n-1)^{d+1}}{n^{d+1}}(\Delta_t)^d\Delta_l, \end{aligned}\tag{5.16}$$

with $\Delta_t = t_n - \mu_{t,\mathcal{Q}_1}$ and $\Delta_l = l_n - \mu_{l,\mathcal{Q}_1}$.

5.3.3 Denominator

The denominator of Equation (5.7) requires the computation of two central sums. For the second central sum $\sum_{i=1}^{n}(l_i - \mu_l)^2$ we already gave pair-wise iterative as well as incremental formulas for $CS_{2,\mathcal{Q}}$ in Equation (3.10) and Equation (4.3). The first central sum $\sum_{i=1}^{n}(t'_i - \mu_{t'})^2$ relates to the variance of the preprocessed traces. For this, we already introduced efficient formulas in the previous chapter. Further, having the formulas given in Section 5.3.2 the correlation of a univariate CPA at any arbitrary order d can be easily derived.

5.4 Multivariate CPA

In the following we give iterative formula for multivariate higher-order CPA with the optimum combination function, i.e., centered product [PRB09, SVO⁺10]. Given d sample point indices $\mathcal{J} = \{j_1, ..., j_d\}$ as the points to be combined and a set of sample vectors $\mathcal{Q} = \{\boldsymbol{V}_{i\in\{1,\dots,n\}}\}$ with $\boldsymbol{V}_i = \left(t_i^{(j)} \mid j \in \mathcal{J}\right)$, the centered product of the ith trace is defined as

$$c_i = \prod_{j\in\mathcal{J}}\left(t_i^{(j)} - \mu_{\mathcal{Q}}^{(j)}\right), \tag{5.17}$$

where $\mu_{\mathcal{Q}}^{(j)}$ denotes the mean at sample point j over set $\mathcal{Q}$.

The authors of [BB16] proposed an iterative formula for the Pearson's correlation coefficient in the bivariate case, i.e., $d = 2$. However, during the computation they calculate the sum $\sum_{i=1}^{n}\left(t_i^{(j_1)}t_i^{(j_2)}\right)^2$ for the two point indices j_1 and j_2 (cf. s_{11} of Table 5 in [BB16]). Their method is basically equivalent to using the raw moments to derive higher-order statistical moments. Given a high number of traces this value can grow very large, and can cause numerical instability.

We instead provide iterative formulas based on mean-free values. In our approach, the formula for the multivariate Pearson's correlation coefficient is first simplified using Equation (5.7) to

$$\rho = \frac{\frac{1}{n}\sum_{i=1}^{n}(c_i-\mu_c)(l_i-\mu_l)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(c_i-\mu_c\right)^2\frac{1}{n}\sum_{i=1}^{n}\left(l_i-\mu_l\right)^2}} = \frac{\frac{1}{n}\sum_{i=1}^{n}\Big(c_i(l_i-\mu_l)\Big)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(c_i-\mu_c\right)^2\frac{1}{n}\sum_{i=1}^{n}\left(l_i-\mu_l\right)^2}}. \tag{5.18}$$

5.4.1 Numerator

The way of computing the numerator of Equation (5.18)

$$\frac{1}{n}\sum_{i=1}^{n}\Big(c_i(l_i-\mu_l)\Big) = \frac{1}{n}\sum_{i=1}^{n}\Big(\prod_{j\in\mathcal{J}}\left(t_i^{(j)}-\mu_{\mathcal{Q}}^{(j)}\right)(l_i-\mu_l)\Big) \tag{5.19}$$

is similar to the iterative computation of the first parameter for the multivariate t-test. We indeed can write Equation (5.19) as

$$\frac{1}{n}\sum_{i=1}^{n}\Big(c_i(l_i-\mu_l)\Big) = \frac{1}{n}\sum_{i=1}^{n}\prod_{j\in\mathcal{J}'}\left(t_i^{(j)}-\mu_{\mathcal{Q}}^{(j)}\right), \tag{5.20}$$

with $\mathcal{J}'=\mathcal{J}\cup\{j^*\}$, $t_i^{(j*)}=l_i$ and $\mu_{\mathcal{Q}}^{(j*)}=\mu_l$. Equation (4.12) relates to the incremental case when set $\mathcal{Q}_2$ has a cardinality of 1. Below we present a generalization of this method to arbitrary sized $\mathcal{Q}_2$.

Generalization of Theorem 4.5.1

Theorem 5.4.1. *Let $\mathcal{J}'$ be a given set of indices (of $d+1$ points of interest) and two sets of sample vectors $\mathcal{Q}_1=\{\boldsymbol{V}_{i\in\{1,\dots,n_1\}}\}$, $\mathcal{Q}_2=\{\boldsymbol{V}_{i\in\{1,\dots,n_2\}}\}$ with $\boldsymbol{V}_i=\left(t_i^{(j)}\mid j\in\mathcal{J}'\right)$. The sum of the centered products $SCP_{d+1,\mathcal{Q},\mathcal{J}'}$ of the extended set $\mathcal{Q}=\mathcal{Q}_1\cup\mathcal{Q}_2$ with $\Delta^{(j\in\mathcal{J}')}=\mu_{\mathcal{Q}_2}^{(j)}-\mu_{\mathcal{Q}_1}^{(j)}$ and $|\mathcal{Q}|=n$ can be computed as:*

$$\begin{aligned} SCP_{d+1,\mathcal{Q},\mathcal{J}'} &= SCP_{d+1,\mathcal{Q}_1,\mathcal{J}'}+SCP_{d+1,\mathcal{Q}_2,\mathcal{J}'} \\ &\quad+\sum_{b=2}^{d}\sum_{\mathcal{S}\in\mathcal{P}_b}\left((-n_2)^{d+1-b}SCP_{b,\mathcal{Q}_1,\mathcal{S}}+n_1^{d+1-b}SCP_{b,\mathcal{Q}_2,\mathcal{S}}\right)\prod_{j\in\mathcal{J}'\setminus\mathcal{S}}\frac{\Delta^{(j)}}{n} \\ &\quad+\frac{(-n_2)^{d+1}n_1+n_1^{d+1}n_2}{n^{d+1}}\prod_{j\in\mathcal{J}'}\Delta^{(j)}. \end{aligned} \tag{5.21}$$

Proof. We start with the definition of the sum of the centered product.

$$\begin{aligned} SCP_{d+1,\mathcal{Q},\mathcal{J}'} &= \sum_{\boldsymbol{V}\in\mathcal{Q}}\prod_{j\in\mathcal{J}'}\left(t^{(j)}-\mu_{\mathcal{Q}}^{(j)}\right) \\ &= \sum_{\boldsymbol{V}\in\mathcal{Q}_1}\prod_{j\in\mathcal{J}'}\left(t^{(j)}-\mu_{\mathcal{Q}}^{(j)}\right)+\sum_{\boldsymbol{V}\in\mathcal{Q}_2}\prod_{j\in\mathcal{J}'}\left(t^{(j)}-\mu_{\mathcal{Q}}^{(j)}\right) \end{aligned} \tag{5.22}$$

Using $\mu_{\mathcal{Q}}^{(j)} = \dfrac{n_1\mu_{\mathcal{Q}_1}^{(j)} + n_2\mu_{\mathcal{Q}_2}^{(j)}}{n}$ and $\Delta^{(j\in\mathcal{J}')} = \mu_{\mathcal{Q}_2}^{(j)} - \mu_{\mathcal{Q}_1}^{(j)}$, we rewrite the first sum of Equation (5.22) as

$$
\begin{aligned}
\sum_{\boldsymbol{V}\in\mathcal{Q}_1}\prod_{j\in\mathcal{J}'}\left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)}\right) &= \sum_{\boldsymbol{V}\in\mathcal{Q}_1}\prod_{j\in\mathcal{J}'}\left(t^{(j)} - \frac{n_1\mu_{\mathcal{Q}_1}^{(j)} + n_2\mu_{\mathcal{Q}_2}^{(j)}}{n}\right)\\
&= \sum_{\boldsymbol{V}\in\mathcal{Q}_1}\prod_{j\in\mathcal{J}'}\left(t^{(j)} - \mu_{\mathcal{Q}_1}^{(j)} - \frac{n_2}{n}\Delta^{(j)}\right)\\
&= \left(\sum_{\boldsymbol{V}\in\mathcal{Q}_1}\prod_{j\in\mathcal{J}'}\left(t^{(j)} - \mu_{\mathcal{Q}_1}^{(j)}\right)\right)\\
&+ \left(\sum_{b=1}^{d}\sum_{\mathcal{S}\in\mathcal{P}_b}\sum_{\boldsymbol{V}\in\mathcal{Q}_1}\prod_{s\in\mathcal{S}}\left(t^{(s)} - \mu_{\mathcal{Q}_1}^{(s)}\right)\prod_{j\in\mathcal{J}'\setminus\mathcal{S}}\frac{n_2\Delta^{(j)}}{-n}\right)\\
&+ \left(\sum_{\boldsymbol{V}\in\mathcal{Q}_1}\prod_{j\in\mathcal{J}'}\frac{n_2\Delta^{(j)}}{-n}\right).
\end{aligned}
\tag{5.23}
$$

With Equation (4.10) and the fact that $\forall\, j \in \mathcal{J}'$, $\sum\limits_{\boldsymbol{V}\in\mathcal{Q}_1}\left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)}\right) = 0$, we can simplify Equation (5.23) to

$$
\begin{aligned}
\sum_{\boldsymbol{V}\in\mathcal{Q}_1}\prod_{j\in\mathcal{J}'}\left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)}\right) &= SCP_{d+1,\mathcal{Q}_1,\mathcal{J}'} + \left(\sum_{b=2}^{d}\sum_{\mathcal{S}\in\mathcal{P}_b} SCP_{b,\mathcal{Q}_1,\mathcal{S}}\prod_{j\in\mathcal{J}'\setminus\mathcal{S}}\frac{n_2\Delta^{(j)}}{-n}\right)\\
&+ \frac{(-n_2)^{d+1}\,n_1}{n^{d+1}}\prod_{j\in\mathcal{J}'}\Delta^{(j)}.
\end{aligned}
\tag{5.24}
$$

By following the same procedure we can write the second sum of Equation (5.22) as

$$
\begin{aligned}
\sum_{\boldsymbol{V}\in\mathcal{Q}_2}\prod_{j\in\mathcal{J}'}\left(t^{(j)} - \mu_{\mathcal{Q}}^{(j)}\right) &= SCP_{d+1,\mathcal{Q}_2,\mathcal{J}'} + \left(\sum_{b=2}^{d}\sum_{\mathcal{S}\in\mathcal{P}_b} SCP_{b,\mathcal{Q}_2,\mathcal{S}}\prod_{j\in\mathcal{J}'\setminus\mathcal{S}}\frac{n_1\Delta^{(j)}}{n}\right)\\
&+ \frac{n_1^{d+1}n_2}{n^{d+1}}\prod_{j\in\mathcal{J}'}\Delta^{(j)}.
\end{aligned}
\tag{5.25}
$$

The combination of (5.24) and (5.25) results in the iterative formula given in Equation (5.21). □

5.4.2 Denominator

Similar to the expressions given in Section 5.3.3 the denominator of Equation (5.18) consists of two central sums. The second one $\sum\limits_{i=1}^{n}(l_i - \mu_l)^2$ is the same as that of the univariate CPA and Equation (3.10) and Equation (4.3) are still valid.

For the first central sum $\sum\limits_{i=1}^{n}\left(c_i-\mu_c\right)^2$ we recall the formulas for the estimation of the variance of the preprocessed traces in a multivariate setting. It means that we can write

$$\begin{aligned}\sum_{i=1}^{n}\left(c_i-\mu_c\right)^2 &= \sum_{\boldsymbol{V}\in\mathcal{Q}}\left(\prod_{j\in\mathcal{J}}\left(t^{(j)}-\mu_{\mathcal{Q}}^{(j)}\right)-\frac{SCP_{d,\mathcal{Q},\mathcal{J}}}{n}\right)^2 \\ &= SCP_{2d,\mathcal{Q},\mathcal{J}''}-\frac{\left(SCP_{d,\mathcal{Q},\mathcal{J}}\right)^2}{n}, \qquad (5.26)\end{aligned}$$

with multiset $\mathcal{J}''=\{j_1,...,j_d,j_1,...,j_d\}$. It is noteworthy that in contrast to the computation of the numerator, where the set $\mathcal{J}'$ with $d+1$ indices is used, here for the denominator the set $\mathcal{J}$ and its extension $\mathcal{J}''$ with respectively d and $2d$ indices are applied.

5.5 Moments-Correlating DPA

MC-DPA [MS14] as a successor of Correlation-Enhanced Power Analysis Collision Attack (CEPACA) [MME10] solves its shortcomings and is based on correlating the moments to the traces [MI14, DFS15, DSV$^+$15]. It relaxes the necessity of a hypothetical leakage model which is essential in the case of a CPA.

The most general form of MC-DPA is Moments-Correlating Profiled DPA (MCP-DPA). In such a scenario, the traces used to build the model $\boldsymbol{t}^{(M)}_{i\in\{1,\ldots,n^{(M)}\}}$ (and trivially their number $n^{(M)}$) are not necessarily the same as the traces used in the attack $\boldsymbol{t}_{i\in\{1,\ldots,n\}}$. An MC-DPA in a multivariate settings uses two sets of sample point indices $\mathcal{J}_M$ and $\mathcal{J}_t$ related to the sample points of the model and the attack respectively. Such sample points are taken based on the time instances when a certain function (e.g., an Sbox) operates on an intermediate value $v^{(M)}_{i\in\{1,\ldots,n^{(M)}\}}$ to form the model and on another intermediate value $v^{(t)}_{i\in\{1,\ldots,n\}}$ to perform the attack. In a simple scenario, such intermediate values can be different Sbox inputs. Optionally, a leakage function can be considered as $\widetilde{\mathsf{L}}(.)$ over the targeted intermediate values. Note that in the most general form such a leakage function can be the identity mapping, i.e., $\widetilde{\mathsf{L}}(v)=v$. Following the original MC-DPA scheme [MS14], $v_i^{(M)}=d_i^{(M)}\oplus k^{(M)}$ and $v_i^{(t)}=d_i^{(t)}\oplus k^{(t)}$ with $d^{(M)}$ and $d^{(t)}$ e.g., plaintext portions (bytes) respectively of the model and the attack. Hence, due to the linear relations such a setting turns into a linear collision attack [Bog08] with $\widetilde{\mathsf{L}}(v_i^{(M)})=d_i^{(M)}$ and $\widetilde{\mathsf{L}}(v_i^{(t)})=d_i^{(t)}\oplus\Delta k$, which is referred to as Moments-Correlating Collision DPA (MCC-DPA), where the traces for the model and the attack are the same and $n^{(M)}=n$. However, in the following expressions we consider the profiling one which can be easily simplified to the collision one.

Let us denote $\mathcal{L}$ as a set of all possible outputs of the leakage function with cardinality of $n_{\mathcal{L}}$ is defined as

$$\mathcal{L}=\{l^{(1)},\ldots,l^{(n_{\mathcal{L}})}\}=\{l\,|\,\exists v,\,\widetilde{\mathsf{L}}(v)=l\}. \qquad (5.27)$$

Correspondingly we define $n_{\mathcal{L}}$ subsets $\mathcal{I}^{(M)}_{l^{(a\in\{1,\ldots,n_{\mathcal{L}}\})}}$

$$\mathcal{I}^{(M)}_{l^{(a)}}=\{i\in\{1,\ldots,n^{(M)}\}\;|\widetilde{\mathsf{L}}(v_i^{(M)})=l^{(a)}\} \qquad (5.28)$$

as the trace indices with particular leakage value $l^{(a)}$ on the model's intermediate values $v_i^{(M)}$ with cardinality of $n_{l^{(a)}}^{(M)}$. The same subsets are also defined with respect to the attack's intermediate values $v_i^{(t)}$as

$$\mathcal{I}_{l^{(a)}}^{(t)} = \{i \in \{1, \dots, n\} \ |\widetilde{\mathsf{L}}(v_i^{(t)}) = l^{(a)}\}, \tag{5.29}$$

with $|\mathcal{I}_{l^{(a)}}^{(t)}| = n_{l^{(a)}}^{(t)}$.

Depending on the type of the attack (univariate vs. multivariate) the sample points at $\mathcal{J}_M$ are first combined using a combining function, e.g., centered product, split into the subsets depending on the leakage model $\widetilde{\mathsf{L}}(.)$ and then used to estimate the statistical moments of a given order d. Depending on the order of the attack, further preprocessing is also necessary. We denote these moments as the model by

$$\forall l^{(a)} \in \mathcal{L},\ M_{l^{(a)}} \xleftarrow{\substack{\text{preprocessing,}\\ \text{(central/standardized)}\\ \text{d-order moment}}} \{\boldsymbol{t}_i^{(M)},\, i \in \mathcal{I}_{l^{(a)}}^{(M)}, \mathcal{J}_M\}. \tag{5.30}$$

On the other hand, the traces at the sample points $\mathcal{J}_t$ need also to be preprocessed according to the variate of the attack (univariate vs. multivariate) as well as the given order d.

The correlation between the moments $M_{l^{(a \in \{1,\dots,n_{\mathcal{L}}\})}}$ and the preprocessed traces $t'_{i \in \{1,\dots,n\}}$ is defined as

$$\rho = \frac{\frac{1}{n} \sum\limits_{i=1}^{n} (t'_i - \mu_{t'})(M_{l_i} - \mu_M)}{\sqrt{\frac{1}{n} \sum\limits_{i=1}^{n} (t'_i - \mu_{t'})^2 \ \frac{1}{n} \sum\limits_{i=1}^{n} (M_{l_i} - \mu_M)^2}}, \tag{5.31}$$

where $M_{l_{i \in \{1,\dots,n\}}} = M_{l^{(a)}}$, $l^{(a)} = \widetilde{\mathsf{L}}(v_i^{(t)}) \in \mathcal{L}$.

5.5.1 Numerator

To compute the numerator of Equation (5.31) it is first simplified to

$$\frac{1}{n} \sum_{i=1}^{n} (t'_i - \mu_{t'})(M_{l_i} - \mu_M) = \sum_{a=1}^{n_{\mathcal{L}}} (M_{l^{(a)}} - \mu_M) \frac{1}{n} \sum_{i \in \mathcal{I}_{l^{(a)}}^{(t)}} t'_i. \tag{5.32}$$

The preprocessing of the MC-DPA requires the sum of Equation (5.32) $SUM_{\mathcal{I}_{l^{(a)}}^{(t)}} = \sum\limits_{i \in \mathcal{I}_{l^{(a)}}^{(t)}} t'_i$ to be processed independently. Otherwise, it is not trivially possible to provide iterative formulas as the mean and variance of subgroup of the traces $\in \mathcal{I}_{l^{(a)}}^{(t)}$ change. Since $n_{\mathcal{L}}$ is limited, we store a sum for each value of set $\mathcal{L}$ and merge them only at the end when the value of the estimated correlation is desired. In the multivariate higher-order $d > 1$ scenario, we store $n_{\mathcal{L}}$ sums of the traces as

$$SUM_{\mathcal{I}_{l^{(a)}}^{(t)}} = \sum_{i \in \mathcal{I}_{l^{(a)}}^{(t)}} t'_i = \sum_{i \in \mathcal{I}_{l^{(a)}}^{(t)}} \prod_{j \in \mathcal{J}_t} \left(t_i^{(j)} - \mu_{\mathcal{I}_{l^{(a)}}^{(t)}}^{(j)} \right) = SCP_{d, \mathcal{I}_{l^{(a)}}^{(t)}, \mathcal{J}_t}, \tag{5.33}$$

and in case of the univariate higher-order $d > 2$ as

$$SUM_{\mathcal{I}^{(t)}_{l^{(a)}}} = \sum_{i \in \mathcal{I}^{(t)}_{l^{(a)}}} t'_i = \frac{1}{\left(\sigma_{\mathcal{I}^{(t)}_{l^{(a)}}}\right)^d} \sum_{i \in \mathcal{I}^{(t)}_{l^{(a)}}} \left(t_i - \mu_{\mathcal{I}^{(t)}_{l^{(i)}}}\right)^d = \frac{1}{\left(\sigma_{\mathcal{I}^{(t)}_{l^{(i)}}}\right)^d} CS_{d,\mathcal{I}^{(t)}_{l^{(i)}}} . \tag{5.34}$$

Note that for $d = 2$ the denominator of Equation (5.34) is omitted. For a univariate first-order attack the means are used to derive the latter term of Equation (5.32) as

$$\frac{1}{n} SUM_{\mathcal{I}^{(t)}_{l^{(a)}}} = \frac{1}{n} \sum_{i \in \mathcal{I}^{(t)}_{l^{(a)}}} t_i = \frac{n^{(t)}_{l^{(a)}}}{n} \mu_{\mathcal{I}^{(t)}_{l^{(a)}}} . \tag{5.35}$$

We should here emphasize that – in contrast to the methods of the prior sections – in case of MC-DPA when a new trace is added to the set of traces following the incremental formulas only the sum and the moments which correspond to the leakage value $l^{(a)}$ related to the new trace are updated.

In order to calculate the whole numerator it is necessary to store the moments $M_{l^{(a)}}, \forall l^{(a)} \in \mathcal{L}$. This procedure is similar to before, and for the multivariate higher-order case it can be done by computing

$$M_{l^{(a)}} = \frac{1}{n^{(M)}_{l^{(a)}}} \sum_{i \in \mathcal{I}^{(M)}_{l^{(a)}}} \prod_{j \in \mathcal{J}_M} \left(t_i^{(j)} - \mu^{(j)}_{\mathcal{I}^{(M)}_{l^{(a)}}}\right) = \frac{SCP_{d,\mathcal{I}^{(M)}_{l^{(a)}},\mathcal{J}_M}}{n^{(M)}_{l^{(a)}}} . \tag{5.36}$$

For the univariate case Equation (5.36) changes analog Equation (5.34). In a univariate first-order attack there is no preprocessing, and $M_{l^{(a)}}$ simply represents the mean $\mu_{\mathcal{I}^{(M)}_{l^{(a)}}}$.

The mean μ_M in Equation (5.31) is

$$\mu_M = \frac{1}{n} \sum_{a=1}^{n_{\mathcal{L}}} n^{(t)}_{l^{(a)}} M_{l^{(a)}} , \tag{5.37}$$

and as an example in case of a multivariate higher-order attack can be written as

$$\mu_M = \frac{1}{n} \sum_{a=1}^{n_{\mathcal{L}}} SCP_{d,\mathcal{I}^{(t)}_{l^{(i)}},\mathcal{J}_M} . \tag{5.38}$$

Since the iterative formulas (for both pair-wise and incremental cases) to compute $SCP_{d,\ldots}$ and $CS_{d,\ldots}$ as well as other necessary moments are given in previous sections, the numerator of Equation (5.31) can be easily derived.

5.5.2 Denominator

The first part of the denominator can be written as

$$\frac{1}{n} \sum_{i=1}^{n} \left(t'_i - \mu_{t'}\right)^2 = \frac{1}{n} \sum_{i=1}^{n} t'^2_i - \left(\mu_{t'}\right)^2 = \frac{1}{n} \sum_{a=1}^{n_{\mathcal{L}}} \left(\sum_{i \in \mathcal{I}^{(t)}_{l^{(a)}}} t'^2_i \right) - \left(\mu_{t'}\right)^2 . \tag{5.39}$$

Therefore, we additionally need to compute the sums of the squared preprocessed traces $SUM^2_{\mathcal{I}^{(t)}_{l^{(a)}}} = \sum\limits_{i \in \mathcal{I}^{(t)}_{l^{(a)}}} t_i'^2$. For a multivariate higher-order case, this is written as $SCP_{2d,\mathcal{I}^{(t)}_{l^{(a)}},\{\mathcal{J}^t,\mathcal{J}^t\}}$ similar to Equation (5.33) or similar to Equation (5.34) and Equation (5.35) for the univariate cases. Further, the sums $SUM_{\mathcal{I}^{(t)}_{l^{(a)}}}$ computed by Equation (5.33), Equation (5.34), or Equation (5.35) can be used to derive $\mu_{t'}$ following the same principle of Equation (5.37).

The second part of the denominator of Equation (5.31) can be obtained from the values that are already used to compute the numerator:

$$\frac{1}{n}\sum_{i=1}^{n}(M_{l_i} - \mu_M)^2 = \frac{1}{n}\sum_{a=1}^{n_\mathcal{L}} n^{(t)}_{l^{(a)}}(M_{l^{(a)}} - \mu_M)^2. \tag{5.40}$$

Since $n_\mathcal{L}$ is limited, the above expression can be computed at the end when all traces are processed to estimate the correlation.

In the aforementioned approach the sums $SUM_{\mathcal{I}^{(t)}_{l^{(a)}}}$ are grouped based on the output of the leakage function, i.e., $l^{(a)}$, which is also key dependent. Hence, the traces have to be regrouped for each key candidate as well as for each selected leakage function $\widetilde{\mathsf{L}}(.)$.

5.5.3 Reuse of Sums

In the aforementioned approach the sums $SUM_{\mathcal{I}^{(t)}_{l^{(a)}}}$ are grouped based on the output of the leakage function, i.e., $l^{(a)}$, which is also key dependent. Hence, the traces have to be regrouped for each key candidate as well as for each selected leakage function $\widetilde{\mathsf{L}}(.)$. Below we provide an alternative approach that enables reuse of sums and avoids the recomputations for each key candidate or each leakage function.

If $\widetilde{\mathsf{L}}(.)$ is a one-to-one mapping (bijection), reuse of the sums $SUM_{\mathcal{I}^{(t)}_{l^{(a)}}}$ is trivial as each possible value of the corresponding associated data (e.g., plaintext byte) maps to a different $l^{(a)}$. Thus, it is sufficient to use different mapping of leakage values in Equation (5.32) and all computed sums can be reused as they are.

If the leakage function is a many-to-one mapping, reuse of sums would require splitting each sum into parts depending on the associated data, and then merge the sums based on the selected key candidate and underlying leakage function after processing all traces. Therefore, the procedures described in Section 5.5.1 and Section 5.5.2 need to be adapted to enable reuse of such sums.

To this end, we first denote $\mathcal{D}$ as the set of all possible associated data parts (related either to the model $d^{(M)}$ or to the attack $d^{(t)}$ with cardinality of $n_\mathcal{D}$ as

$$\mathcal{D} = \{d^{(1)}, \ldots, d^{(n_\mathcal{D})}\} = \Big\{ d \,\big|\, \exists i,\, d^{(M)}_{i \in \{1,\ldots,n\}} = d \vee d^{(t)}_{i \in \{1,\ldots,n^{(M)}\}} = d \Big\}. \tag{5.41}$$

Hence, instead of grouping the traces based on the leakage values, we classify them based on the associated data values $d^{(M)}$ or $d^{(t)}$. Since $n_\mathcal{L} \leq n_\mathcal{D}$, this approach can be problematic if $n_\mathcal{D}$ is too big. However, in common scenarios (e.g., $n_\mathcal{D} = 256$ for 8-bit associated data parts) this should not cause any problem unless a leakage function over consecutive intermediate values, e.g., a Hamming-distance model, is desired.

Now instead of $\mathcal{I}^{(t)}_{d^{(e)}}$ (resp. $\mathcal{I}^{(M)}_{d^{(e)}}$) we define subsets of the indices

$$\mathcal{I}^{(t)}_{d^{(e\in\{1,\dots,n_{\mathcal{D}}\})}} = \left\{ i \in \{1,\dots,n\} \,|\, d^{(t)}_i = d^{(e)} \right\}, \tag{5.42}$$

with the cardinality of $n^{(t)}_{d^{(e)}}$ (resp. $n^{(M)}_{d^{(e)}}$). Suppose that such subsets are used to compute the sums, e.g., $SUM_{\mathcal{I}^{(t)}_{d^{(e)}}}$, in one of Equation (5.32) - Equation (5.35) as well as in Equation (5.39) depending on the variate as well as the order of the attack.

Given a key candidate and a leakage function we define the following $n_{\mathcal{L}}$ small sets

$$\mathcal{E}^{(t)}_{l^{(a\in\{1,\dots,n_{\mathcal{L}}\})}} = \left\{ e \in \{1,\dots,n_{\mathcal{D}}\} \,|\, \widetilde{\mathsf{L}}\left(v^{(t)} = \mathsf{F}\left(d^{(e)}, k^{(t)}\right)\right) = l^{(a)} \right\}, \tag{5.43}$$

with the cardinality of $n^{(t)}_{\mathcal{E}_{l^{(a)}}}$ as the indices of the corresponding data (of the attack) $\in \mathcal{D}$ which map to the same leakage. The same is defined for the model as

$$\mathcal{E}^{(M)}_{l^{(a\in\{1,\dots,n_{\mathcal{L}}\})}} = \left\{ e \in \{1,\dots,n_{\mathcal{D}}\} \,|\, \widetilde{\mathsf{L}}\left(v^{(M)} = \mathsf{F}\left(d^{(e)}, k^{(M)}\right)\right) = l^{(a)} \right\}, \tag{5.44}$$

with $n^{(M)}_{\mathcal{E}_{l^{(a)}}} = |\mathcal{E}^{(M)}_{l^{(a)}}|$. Respectively $k^{(t)}$ and $k^{(M)}$ denote the part of the (guessed) key of the attack and the model.

In a first-order univariate attack the regrouping is easily possible as it does not involve any preprocessing. To calculate $SUM_{\mathcal{I}^{(t)}_{l^{(a)}}} = \sum_{i\in\mathcal{I}^{(t)}_{l^{(a)}}} t_i$, $a \in \{1,\dots,n_{\mathcal{L}}\}$ and the corresponding cardinality $n^{(t)}_{l^{(a)}}$ we combine the computed sums and cardinalities as

$$SUM_{\mathcal{I}^{(t)}_{l^{(a)}}} = \sum_{e\in\mathcal{E}^{(t)}_{l^{(a)}}} \left(\sum_{i\in\mathcal{I}^{(t)}_{d^{(e)}}} t_i \right) = \sum_{e\in\mathcal{E}^{(t)}_{l^{(a)}}} SUM_{\mathcal{I}^{(t)}_{d^{(e)}}}, \qquad n^{(t)}_{l^{(a)}} = \sum_{e\in\mathcal{E}^{(t)}_{l^{(a)}}} n^{(t)}_{d^{(e)}}. \tag{5.45}$$

The same procedure needs to be repeated for the model (i.e., means in case of a first-order univariate attack) as

$$M_{l^{(a)}} = \mu_{\mathcal{I}^{(M)}_{l^{(a)}}} = \frac{1}{n^{(M)}_{l^{(a)}}} \sum_{e\in\mathcal{E}^{(M)}_{l^{(a)}}} n^{(M)}_{d^{(e)}}\, \mu_{\mathcal{I}^{(M)}_{d^{(e)}}}, \qquad n^{(M)}_{l^{(a)}} = \sum_{e\in\mathcal{E}^{(M)}_{l^{(a)}}} n^{(M)}_{d^{(e)}}. \tag{5.46}$$

The first part of the denominator of Equation (5.31) as the variance of the traces is independent of the key guesses as well as the leakage function if the MC-DPA is a first-order univariate attack. Hence, the regrouping the traces for the first part of the denominator is not necessary, and one can derive such a value by the sums computed based on the associated data $SUM_{\mathcal{I}^{(t)}_{d^{(e)}}}$.

In higher-order scenarios, this method cannot be easily applied because of the preprocessing which depends on the leakage function. To this end, the sums $SUM_{\mathcal{I}^{(t)}_{d^{(e)}}}$ need to be adjusted to enable the regrouping. For that, we first compute the means $\mu_{\mathcal{I}^{(t)}_{l^{(a)}}}$ for each $l^{(a)} \in \mathcal{L}$ similar to Equation (5.46). In a d-order univariate case ($d > 1$) the $n_{\mathcal{D}}$ sums are then adjusted to be preprocessed with these means as ($e \in \{1, \dots, n_{\mathcal{D}}\}$)

$$\begin{aligned} SUM_{\mathcal{I}^{(t)}_{d^{(e)}}} = \sum_{i \in \mathcal{I}^{(t)}_{d^{(e)}}} \left(t_i - \mu_{\mathcal{I}^{(t)}_{l^{(a)}}}\right)^d &= \sum_{i \in \mathcal{I}^{(t)}_{d^{(e)}}} \left(t_i - \mu_{\mathcal{I}^{(t)}_{d^{(e)}}} + \Delta\right)^d \\ &= CS_{d,\mathcal{I}^{(t)}_{d^{(e)}}} + \sum_{p=1}^{d-2} \binom{p}{d} CS_{d-p,\mathcal{I}^{(t)}_{d^{(e)}}} \Delta^p + \Delta^d, \end{aligned} \tag{5.47}$$

with $\Delta = \mu_{\mathcal{I}^{(t)}_{l^{(a)}}} - \mu_{\mathcal{I}^{(t)}_{d^{(e)}}}$, and $l^{(a)} = \widetilde{\mathsf{L}}\left(v^{(t)} = \mathsf{F}\left(d^{(e)}, k^{(t)}\right)\right)$ (see Equation (5.43)).

For multivariate attacks the above equation is similarly defined as

$$\begin{aligned} SUM_{\mathcal{I}^{(t)}_{d^{(e)}}} = \sum_{i \in \mathcal{I}^{(t)}_{d^{(e)}}} \prod_{j \in \mathcal{J}_t} \left(t_i^{(j)} - \mu^{(j)}_{\mathcal{I}^{(t)}_{l^{(a)}}}\right) &= \sum_{i \in \mathcal{I}^{(t)}_{d^{(e)}}} \prod_{j \in \mathcal{J}_t} \left(t_i^{(j)} - \mu^{(j)}_{\mathcal{I}^{(t)}_{d^{(e)}}} + \Delta\right) \\ &= SCP_{d,\mathcal{I}^{(t)}_{d^{(e)}},\mathcal{J}_t} + \prod_{j \in \mathcal{J}} \Delta^{(j)} \\ &+ \left(\sum_{b=2}^{d-1} \sum_{\mathcal{S} \in \mathcal{P}_b} SCP_{b,\mathcal{I}^{(t)}_{d^{(e)}},\mathcal{J}_t} \prod_{j \in \mathcal{J} \setminus \mathcal{S}} \Delta^{(j)}\right), \end{aligned} \tag{5.48}$$

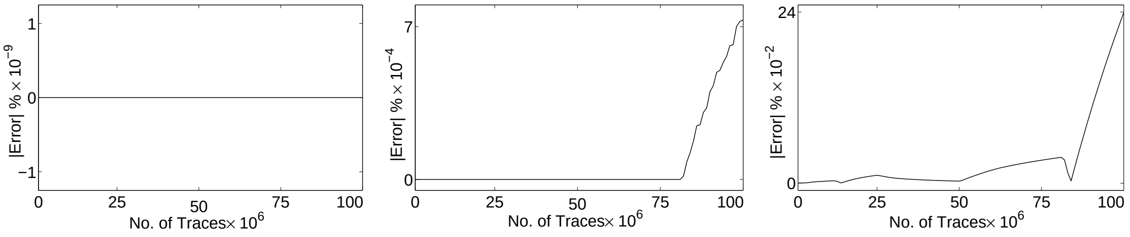
with $\Delta^{(j \in \mathcal{J}_t)} = \mu^{(j)}_{\mathcal{I}^{(t)}_{l^{(a)}}} - \mu^{(j)}_{\mathcal{I}^{(t)}_{d^{(e)}}}$. This adjustment has to be applied to the moments and sums of the square of the preprocessed traces as well which are necessary for the first part of the denominator of Equation (5.31). This can be done similar to Equations (5.47) and Equation (5.48).

Using the above given equations (for any type of MC-DPA) it is now possible to compute the sums based on only the associated data $\mathcal{I}^{(t)}_{d^{(e)}}$ and $\mathcal{I}^{(M)}_{d^{(e)}}$, and with a little computation overhead combine them based on the selected leakage function and each key guess.

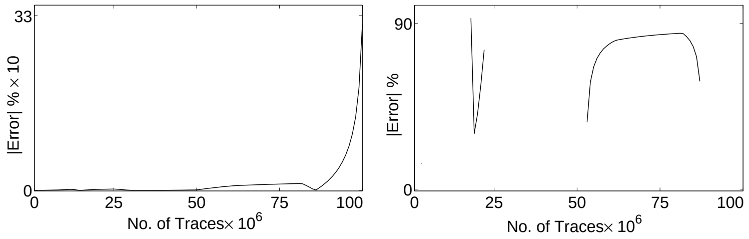
Note that this trick of reusing the sums $SUM_{\mathcal{I}^{(t)}_{d^{(e)}}}$ is not exclusive to MC-DPA. By adjusting the algorithm for all kinds of CPA (univariate, multivariate, first- and higher-orders) this procedure can be applied to each use case discussed in this chapter.

5.6 Evaluation

We evaluate the accuracy (convergence) of our presented approaches, and compare it to the corresponding results of the raw-moment and three-pass approaches. To this end, we generate 100 million simulated leakages by $\sim \mathcal{N}(100 + \mathrm{HW}(x), 3)$, where x is drawn uniformly from $\{0,1\}^4$. Hence, the correlation between the leakages and $\mathrm{HW}(x)$ is estimated. Following the concept of higher-order attacks, the leakages are also preprocessed (up to fifth order) to allow an emulation of a higher-order univariate CPA. Note that the performance results are still valid in the multivariate case given additional leakage points with a similar leakage structure and the normalized product as combination function. This can be easily seen as both type of attacks require the estimation of centralized values up to a power of $2d$ (with an additional standardization for univariate higher-order attacks). The results based on our incremental approaches are exactly the same to the three-pass ones, i.e., with absolute no difference.



(a) 1st-order (b) 2nd-order (c) 3rd-order



(d) 4th-order (e) 5th-order

Figure 5.1: Difference between the result of correlation estimations (raw-moment versus three-pass).

As [BB16] only includes the formulas for first-order and second-order bivariate CPA, we first transform the bivariate formulas to the univariate second-order case and extend the approach to higher orders. Recall that the correlation for the bivariate second-order attack is computed in [BB16] as

$$\rho = \frac{n\lambda_1 - \lambda_2 s_3}{\sqrt{n\lambda_3 - {\lambda_2}^2}\sqrt{ns_9 - {s_3}^2}}, \tag{5.49}$$

where n denotes the number of traces and $\lambda_{\{1,2,3\}}$ are derived from the sums $s_{\{1,\ldots,13\}}$.

For the univariate second-order correlation, some of these sums are equivalent. Therefore, in this special case it is possible to reduce the number of sums required to be computed. For that, we first denote the d-th order sums as

$$S_d^{(t)} = \sum_{i=1}^{n} t_i^d, \qquad S_d^{(l)} = \sum_{i=1}^{n} l_i^d, \qquad S_d^{(t,l)} = \sum_{i=1}^{n} t_i^d l \tag{5.50}$$

with $s_3 = S_1^{(l)}$ and $s_9 = S_2^{(l)}$. The remaining parameters are then derived as

$$\lambda_1 = S_2^{(t,l)} - 2\frac{S_1^{(t)}S_1^{(t,l)}}{n} + \frac{S_1^{(t)}S_1^{(t)}S_1^{(l)}}{n^2}, \quad \lambda_2 = S_2^{(t)} - \frac{S_1^{(t)}S_1^{(t)}}{n}, \tag{5.51}$$

$$\lambda_3 = S_4^{(t)} - 4\frac{S_1^{(t)}S_3^{(t)}}{n} + 6\frac{S_1^{(t)}S_1^{(t)}S_2^{(t)}}{n^2} - 3\frac{S_1^{(t)}S_1^{(t)}S_1^{(t)}S_1^{(t)}}{n^3}. \tag{5.52}$$

For the higher-order correlation the basic structure of Equation (5.49) stays the same, and only the formulas for $\lambda_{\{1,2,3\}}$ change. These are given in Appendix 12.6.3.

With these formulas we computed the correlation up to the fifth order on an Intel Xeon X5670 using a single thread, and examined the differences with respect to the results of the three-pass approach. Figure 5.1 presents the corresponding results. As expected, in the first-order setting the results are exactly the same, but the differences start to be obvious at higher orders particularly for higher number of traces. It is noteworthy that in the cases where no difference is shown for the fifth-order correlation, one of the variances of the denominator in the raw-moment approach turned to a negative value which indicates the instability of such formulas. With respect to the execution time of each approach, although it depends on the optimization level of the underlying computer code, we report 43 s, 17.8 s, and 11.6 s for three-pass, our incremental, and raw-moment approach respectively to estimate all five correlations at the same time on 100 million leakage points. Obviously, the raw-moment approach is faster than the others due to its lower amount of computations compared to our incremental one.

5.7 Conclusion

To conclude, we presented computation procedures which help to significantly improve the performance of correlation-based evaluations. In particular, these techniques enable a fast and efficient computation of Pearson's correlation coefficient for evaluations at arbitrary orders and we showed that our approaches avoid the shortcomings of previous solutions. The presented results are a foundation for further research in the efficient computation of other side-channel evaluation methodologies, e.g., information-theoretic evaluation.

Chapter 6

Advanced Tools for Side-Channel Leakage Estimation

In this chapter, we present our contributions related to information-theoretic evaluation methodologies for side-channel countermeasures based on [SMSG16a]. The accuracy and the fast convergence of a leakage model are both essential components for the efficiency of side-channel analysis. Thus, for efficient leakage estimation an evaluator is requested to pick a PDF that constitutes the optimal trade-off between both aspects. In the case of parametric estimation, Gaussian templates are a common choice due to their fast convergence, given that the actual leakages follow a Gaussian distribution (as in the case of an unprotected device). In contrast, histograms and kernel-based estimations are examples for non-parametric estimation that are capable to capture any distribution (even that of a protected device) at a slower convergence rate. With this chapter, we aim to enlarge the statistical toolbox of a side-channel evaluator by introducing new PDF estimation tools that fill the gap between both extremes. Our tools are designed for parametric estimation and can efficiently characterize leakages up to the fourth statistical moment. We show that such an approach is superior to non-parametric estimators in contexts where key-dependent information in located in one of those moments of the leakage distribution. Furthermore, we successfully demonstrate how to apply our tools for the (worst-case) information-theoretic evaluation on masked implementations with up to four shares, both in a profiled and non-profiled attack scenario. We like to remark that this flexibility capturing information from different moments of the leakage PDF can provide very valuable feedback for hardware designers to their task to evaluate the individual and combined criticality of leakages in their (protected) implementations.

Contents of this Chapter

6.1 Introduction **64**
6.2 Background **66**
6.3 New Proposals **68**
6.4 Simulated Experiments **72**
6.5 Practical Case Studies **74**
6.6 Conclusion and Future Work **84**

6.1 Introduction

Since side-channel analysis is essentially based on the comparison of key-dependent leakage models with actual measurements, there is a central division between *profiled* and *non-profiled* evaluation tools and attacks [WOS14]. In the first case, the adversary/evaluator is allowed to build an accurate (yet not perfect [DSV14]) model for his target device that generally corresponds to an estimation of the leakage PDF[1]. As depicted in the upper left part of Figure 6.1, Gaussian TA are the most common tool for this purpose [CRR02]. In this (here: exhaustive) approach, one builds a Gaussian model for the leakage of every target intermediate value in the implementation. The main limitation of Gaussian templates is that they are bound to the analysis of the first two moments in a leakage distribution (i.e., unprotected implementations and masking with $d = 2$). According to the state-of-the-art, the canonical way to analyze higher-order masked implementations would be to switch to non-parametric PDF estimation, e.g., based on histograms and kernels. But this comes at the cost of two important drawbacks. First, these tools imply a more complex (hence measurement intensive) estimation problem. Second, they estimate all the statistical moments at once, meaning that one loses the detailed intuition that could be obtained from the separate examination of all moments. Alternatively, one could use the MCP-DPA introduced in [MS14] that suffers from the complementary drawback. Namely, since MCP-DPA is essentially a “per moment” approach, the intuitions extracted now only correspond to moment taken separately, and it is unclear how one could extend these attacks towards the joint exploitation of multiple moments at the same time.

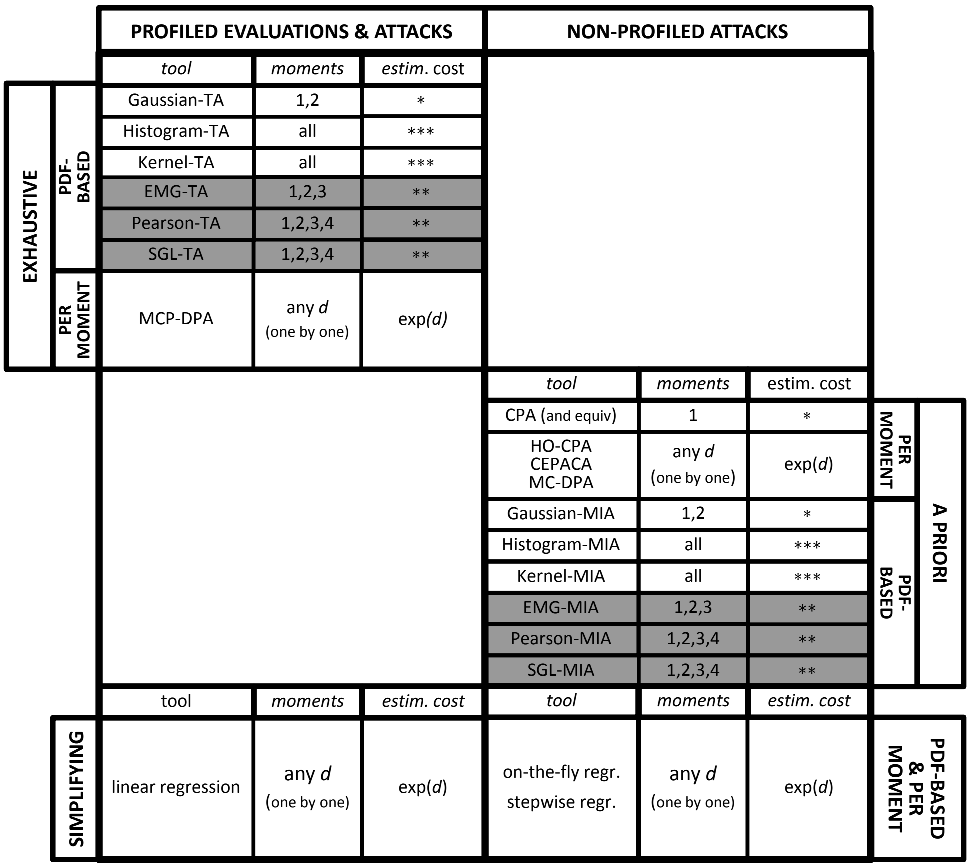
A comprehensive understanding of how the information leakage of a masked cryptographic implementation is spread among different statistical moments is essential to interpret the results of its security evaluation. That is, in general a $(d-1)$th-order secure implementation is defined as an implementation for which the smallest key-dependent moment in the leakage distribution is d, and this is ideally expected to occur for d shares. But in practice, it frequently happens that glitches (i.e., non-independent leakages) contradict this expectation, leading to informative moments of smaller orders than d, both in hardware and software case studies [CGP$^+$12, MPG05]. Significant research efforts have been dedicated to the design of glitch-free implementations, e.g., based on multi-party computation [RP12] or threshold implementations [MPL$^+$11, NRS11]. However, in the latter case the number of shares is larger than the claimed order. This, however, highlights the demand for the ability to determine the exact moment that actually leaks [BGN$^+$14a]. Simple leakage detection tests (e.g., t-test [SM15a]) can be used for this, however they provide only limited information and merely show the existence of leakage (for a more detailed discussion of the limitations of t-test based leakage detection see [DS16]). Eventually, the recent results in [DFS15] showed that by quantifying the informativeness of each statistical moment in a side-channel attack, one can extrapolate the security level of an implementation in function of the noise in its measurements (i.e., a parameter that is typically easier to adapt for HW engineers).

6.1.1 Contribution

Based on this state-of-the art, our contribution is threefold.

---

[1] Profiled attacks can also be referred to when the adversary possesses a device with a biased randomness source (as masks).



PROFILED EVALUATIONS & ATTACKS

		tool	*moments*	*estim.* cost
EXHAUSTIVE	**PDF-BASED**	Gaussian-TA	1,2	*
		Histogram-TA	all	***
		Kernel-TA	all	***
		EMG-TA	1,2,3	**
		Pearson-TA	1,2,3,4	**
		SGL-TA	1,2,3,4	**
	PER MOMENT	MCP-DPA	any *d* (one by one)	exp*(d)*

NON-PROFILED ATTACKS

tool	*moments*	estim. cost		
CPA (and equiv)	1	*	**PER MOMENT**	**A PRIORI**
HO-CPA CEPACA MC-DPA	any *d* (one by one)	exp(*d*)		
Gaussian-MIA	1,2	*	**PDF-BASED**	
Histogram-MIA	all	***		
Kernel-MIA	all	***		
EMG-MIA	1,2,3	**		
Pearson-MIA	1,2,3,4	**		
SGL-MIA	1,2,3,4	**		

	tool	*moments*	*estim. cost*	*tool*	*moments*	*estim. cost*	
SIMPLIFYING	linear regression	any *d* (one by one)	exp(*d*)	on-the-fly regr. stepwise regr.	any *d* (one by one)	exp(*d*)	**PDF-BASED & PER MOMENT**

Figure 6.1: Summary of side-channel evaluation tools and attacks.

First, we extend the evaluation toolbox for profiled side-channel analysis with three new PDF estimation tools, based on Exponentially Modified Gaussian (EMG) distributions, Pearson distribution system and Shifted Generalized Lognormal (SGL) distributions. As illustrated in the upper left part of Figure 6.1, they allow characterizing statistical moments up to the fourth one, which captures all most relevant masked implementations published so far.

Second, we show that these tools enable the computation of the information leakage in each statistical moment of a leakage distribution (up to the fourth one). We further illustrate that based on such computations, we can design efficient attacks that are able to exploit the information in all the leaking moments jointly, and that the efficiency of these attacks is proportional to the sum of the information provided by each moment.

Eventually, we observe that our tools also have applications in the context of non-profiled side-channel analysis, where the adversary assumes some a priori model for his target implementation (e.g., typically Hamming Weight (HW), Hamming Distance (HD)). In this context as well, one can divide existing solutions between "per moment" and "PDF-based" distinguishers (see the middle right part of Figure 6.1). Usual representatives of the first category include CPA [BCO04] or its equivalents [MOS11] for first-order moments, and higher-order DPA [PRB09], CEPACA [Mor12] or Moments Correlating Collision-DPA (MCC-DPA) [MS14] for higher-order moments. The most common representative of the second category is MIA [GBTP08], which usually relies on (non-parametric) histograms or kernels [BGP$^+$11], although any PDF estimation tool is in principle eligible[2]. We show that MIA based on the previously mentioned PDF estimation tools (EMG, Pearson, SGL) leads to interesting efficiency trade-offs for implementations leaking in moments up to four.

The combination of these tools and methods are valuable inputs for the evaluation of the masking countermeasure, since they allow a more accurate understanding of its implementation weaknesses due to glitches (or any other physical default). Furthermore, they are not limited to analysis techniques and also lead to new attacks exploiting a (practically relevant) combination of moments. Eventually, we remark that our results raise relevant questions regarding the so-called simplifying distinguishers in the bottom of Figure 6.1. In this context, the adversary/evaluator does not build a model for every target intermediate value but for a combination of them (or of their bits). All the published simplifying distinguishers (e.g., linear regression in the profiled case [SLP05], its on-the-fly extension [DPRS11] and stepwise regression [WOS14] in the non-profiled case) mix a “per moment” approach [DDP13] with simple (typically Gaussian) PDF estimations. Hence, finding whether one could combine a simplifying distinguisher (that provides useful intuitions regarding the parts of the computations that leak more) with more complex PDF estimation tools as in this chapter (that provide similarly useful intuitions regarding which moments are leaking) remains an interesting open problem.

6.2 Background

Generally, *density estimation* – as a well-studied field in statistics – refers to two major categories, namely non-parametric and parametric methods. Histograms and kernels are among the well-known non-parametric ones, which do not make any assumptions about the form of the distribution and use only the sampled data to estimate the distribution. By contrast, Gaussian density estimation, which is the most popular parametric PDF estimator, assumes a symmetric form for the distribution, and characterizes it based on its (sample) mean and standard deviation only. As mentioned in the introduction, our focus in this paper is side-channel evaluation, which is commonly based on PDF estimation for building the leakage models. In this section, we shortly recall some frequently-applied PDF estimation techniques in the field of side-channel analysis. We only consider a univariate scenario, which is motivated by our experimental case study in Section 6.5, that is based on a threshold implementation in which all the shares are manipulated in parallel.

6.2.1 Histograms

Among the most straightforward techniques to estimate a PDF, one can group the sampled data into (commonly equally-sized) bins. The probability for a given input x can then be given by $Pr[x] = \frac{bin(x)}{n}$, where $bin(x)$ returns the number of samples in the bin to which x belongs, and n indicates the total number of samples.

Although histograms are a non-parametric estimator, the width of the bins (respectively the number of bins) is an important parameter that can significantly influence the resulting PDF.

---

[2] Such as cumulants which are used in [LB10] to estimate the mutual information.

For certain distributions (e.g., Gaussian), there are practical guidelines on how to select such parameters (e.g., Scott's rule [Sco79] and Freedman-Diaconis rule [FD81]). But for distributions that strongly deviate from these assumed forms, the optimal choice of these parameters is unknown.

In our side-channel context, measurements usually correspond to 8-bit data, as the analogue to digital converters which sample the leakages (by means of an oscilloscope) typically have 8-bit effective length. Therefore, the histogram of side-channel leakages can most precisely be estimated with 256 bins. However, by using such a narrow bin width, the number of required samples to fill the bins increases and makes the estimation more data intensive. Hence, the number of bins is commonly selected with respect to the underlying hypothetical model used by the adversary, e.g., 9 in case of Hamming weights for an 8-bit intermediate value [BGP$^+$11]. As histograms make no assumptions about the distribution, the side-channel leakages of all moments are encapsulated and can be exploited.

6.2.2 Kernels

The foundation of kernel-based density estimation is to approximate the PDF with a sum of so-called kernel functions. That is, for each sample point, a kernel function that is centered around this point (l_i) is added to the probability density function. The density for a given input x can then be estimated as:

$$F(x) = \frac{1}{n\,h} \sum_{i=0}^{n-1} K\left(\frac{x - l_i}{h}\right),$$

where h is the bandwidth and $K(.)$ the kernel function. In contrast to histograms, the kernel-based estimation builds a continuous function which can be integrated. This allows for a faster convergence (i.e., with fewer samples) to the real distribution compared to histograms. For the rest, both methods are similar: they are able to capture any moment of a distribution, but cannot differentiate between them.

Concretely, the kernel function should fulfill the property $\int_{-\infty}^{\infty} K(x)dx = 1$. Although there exist many proposals for such functions (e.g., Gaussian and Epanechnikov, see [BGP$^+$11]), the type of kernel has only little influence on the resulting PDF [She04], and the bandwidth h (also called "smoothing parameter") plays a more important role in the precision of the estimation. A common approach to choose the bandwidth is known as Silverman's rule [Sil86], where h is selected as $c\,\sigma\,n^{-1/5}$, with σ the standard deviation of the samples, and the constant c selected based on the chosen kernel function. Recent results [CTO$^+$14] showed that an adaptive procedure (i.e., dynamically altering h) can lead to the best success rates when the PDFs used in a MIA are estimated by a kernel function. In all practical side-channel experiments presented in Section 6.5, where we applied a kernel-based estimator, we used the Gaussian function $K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right)$ with bandwidth $h = 1$ as in all our experiments with different kernel functions and different bandwidths it showed the best result.

6.2.3 Gaussian Density Estimation

In this case, it is assumed that the leakages follow a Gaussian (normal) distribution. Since a Gaussian distribution considers only the first two moments, it generally leads to a more efficient estimation compared to the non-parametric histograms or kernels (as long as the actual distribution is close enough to a Gaussian one). In other words, if the higher (> second) statistical moments of the underlying distribution of the samples are negligible, Gaussian density estimation is going to be extremely efficient. Gaussian Templates and regression-based models are part of the widely-used tools exploiting such an assumption [DSV14].

Gaussian Mixtures.

We mention that yet another approach to PDF estimation for masked implementations would be to consider mixture distributions. As demonstrated in [SVO$^+$10], this solution is especially efficient when the profiling phase assumes the knowledge of the shares. By contrast, it becomes heuristic – since based on the Expectation Maximization (EM) algorithm – if they are not [LP07], which will be our running scenario in this work. In particular, we will consider contexts where the different modes of the mixture distributions are well interleaved (i.e. when the noise is large enough for masking to enforce good security guarantees), which makes the EM algorithm hard(er) to apply and stands in contrast with contexts where the modes can be trivially identified by the adversary (for example see [MKEP12]). That is, our goal is to investigate simple(r) tools that apply to masking when it delivers its promises and are guaranteed to converge without any need to guess about the number of shares in the target device.

6.3 New Proposals

We now describe three alternative parametric distributions that can cover moments up to the fourth one. We discuss their advantages as well as the challenges one may face to set the parameters to use them.

6.3.1 Exponentially Modified Gaussian

Since the Gaussian distribution is symmetric, its skewness is always zero. The EMG is another parametric distribution which additionally includes this first shape parameter. The PDF of such a distribution, that covers the first three moments, is defined by [Gru72]:

$$F(x) = \frac{\lambda_3}{2} e^{\frac{\lambda_3}{2}(2\lambda_1 + \lambda_3\lambda_2^2 - 2x)} erfc\left(\frac{\lambda_1 + \lambda_3\lambda_2^2 - x}{\sqrt{2}\lambda_2}\right), \tag{6.1}$$

where λ_1, λ_2, λ_3 are the parameters of the distribution and $erfc(.)$ refers to the complementary error function defined as:

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2}\, dt.$$

By means of the sample mean μ, standard deviation σ and skewness γ_1 of the data, these three parameters can be estimated as:

$$\lambda_1 = \mu - \sigma\left(\frac{\gamma_1}{2}\right)^{1/3}, \quad \lambda_2^2 = \sigma^2\left(1 - \left(\frac{\gamma_1}{2}\right)^{2/3}\right), \quad \lambda_3 = \frac{1}{\sigma\left(\frac{\gamma_1}{2}\right)^{1/3}}.$$

It should be noted that EMG does not cover symmetric distributions, i.e., $\gamma_1 = 0$. However, it usually causes no issue in practice (and in particular for side-channel attacks) as the estimated skewness is never exactly zero. Nevertheless, if the underlying skewness is zero, the estimated skewness might be very small. These cases can lead to numerical problems, which can be solved by using libraries for higher precision computations or switching to a distribution which covers zero skewness (Gaussian, Pearson). Besides, note that for a negative skewness $\gamma_1 < 0$, the distribution is parameterized with the absolute value $|\gamma_1|$, and then mirrored around the mean.

6.3.2 Pearson Distribution System

The Pearson distribution system is a collection of probability distributions that can be parameterized using the first four moments. In total twelve different distributions (cf. [Pea95, Pea01, Pea16]) are defined in such a way that depending on the estimated moments one type is preferred, and the corresponding PDF estimation technique is applied. In our experiments we noticed that types I, IV and VI (which are presented in detail below) are the only necessary ones. For further descriptions of the other types, the interested reader is referred to the original articles [Pea95, Pea01, Pea16].

To determine the type of distribution and find the parameters for the associated PDF, we first define b_0, b_1, b_2 as:

$$b_0 = \frac{\sigma(4\beta_2 - 3\beta_1)}{10\beta_2 - 12\beta_1 - 18}, \quad b_1 = \frac{\sqrt{\sigma}\gamma_1(\beta_2 + 3)}{10\beta_2 - 12\beta_1 - 18}, \quad b_2 = \frac{2\beta_2 - 3\beta_1 - 6}{10\beta_2 - 12\beta_1 - 18},$$

where $\beta_1 = \gamma_1^2$ (squared skewness) and β_2 denotes the kurtosis. Based on the estimated skewness and kurtosis, the most suited type is selected as follows. If $\kappa_2 = \frac{b_1^2}{4b_0b_2} < 0$, type I is chosen. Otherwise, if κ_2 is in the interval $]0, 1[$, type IV is preferred. In the last case ($\kappa_2 > 1$) type VI is used. (The remaining cases where $\kappa_2 = 0$ and $\kappa_2 = 1$ require different types of distribution but, as previously mentioned, were not encountered in our experiments and are therefore omitted in this section). A visual representation of these type of distributions in function of γ_1 and β_2 is given in Figure 6.2(a).

In order to estimate the type I and VI distributions, it is necessary to find the roots of the quadratic function:

$$f(x) = b_2x^2 + b_1x + b_0, \tag{6.2}$$

denoted as a_1 and a_2 in the following. The rest of the computations are type specific and briefly described in the following.

Type I.

This distribution is a generalization of the beta distribution using four parameters. In this case, Equation (6.2) has two real roots with different signs. We assume without loss of generality that $a_1 < 0 < a_2$ and define:

$$m_1 = \frac{b_1 + a_1}{b_2(a_2 - a_1)}, \qquad m_2 = \frac{-b_1 - a_2}{b_2(a_2 - a_1)}.$$

The PDF is then defined as:

$$F(x) = \frac{\left(\frac{x-\mu-a_1\sqrt{\sigma}}{(a_2-a_1)\sqrt{\sigma}}\right)^{m_1} \left(1 - \frac{x-\mu-a_1\sqrt{\sigma}}{(a_2-a_1)\sqrt{\sigma}}\right)^{m_2}}{B(m_1 + 1, m_2 + 1)(a_2 - a_1)\sqrt{\sigma}}, \tag{6.3}$$

where $B(.,.)$ refers to the beta function. Hence, x is bounded on both sides within $]a_1\sqrt{\sigma}+\mu, a_2\sqrt{\sigma}+\mu[$.

Type IV.

In this case, we first compute the four parameters m_1, m_2, m_3, m_4 as:

$$m_1 = \frac{1}{2b_2},\ m_2 = \frac{2b_1(1-m_1)}{\sqrt{4b_0b_2-b_1^2}},\ m_3 = \sqrt{\frac{(2m_1-2)^3-(2m_1-2)^2}{(2m_1-2)^2+m_2^2}},\ m_4 = \frac{m_3m_2}{2m_1-2}.$$

Then the PDF can be estimated by:

$$F(x) = \frac{e^{-arctan(\frac{x-\mu-m_4\sigma}{m_3\sigma})m_2}}{B(m_1-\frac{1}{2},\frac{1}{2})m_3\sigma}\left|\frac{\Gamma(m_1+\frac{m_2}{2}i)}{\Gamma(m_1)}\right|^2\left(1+\left(\frac{x-\mu-m_4\sigma}{m_3\sigma}\right)^2\right)^{-m_1}, \tag{6.4}$$

where $\Gamma(.)$ denotes the gamma function. In contrast to types I and VI, this distribution is unbounded on both sides and supports x in the interval $[-\infty,+\infty]$.

Type VI.

This distribution is related to the F distribution. In this case, Equation (6.2) has two real roots with the same sign, and we assume without loss of generality that $|a_1| \leq |a_2|$. For this distribution, we first compute m_1 and m_2 as:

$$m_1 = \frac{a_1+b_1}{b_2(a_2-a_1)}, \qquad m_2 = \frac{a_2-a_1}{b_2(a_2-a_1)}.$$

The PDF is then defined as:

$$F(x) = \frac{\left(\frac{x-\mu-a_1\sqrt{\sigma}}{(a_1-a_2)\sqrt{\sigma}}\right)^{m_1}\left(1+\frac{x-\mu-a_1\sqrt{\sigma}}{(a_1-a_2)\sqrt{\sigma}}\right)^{-(m_1+m_2)}}{B\left(m_1+1,m_2-1\right)|a_1-a_2|\sqrt{\sigma}}. \tag{6.5}$$

Depending on the sign of the skewness, the covered range for x is either $]a_1\sqrt{\sigma}+\mu,+\infty]$ ($\gamma_1 > 0$) or $[-\infty, a_1\sqrt{\sigma}+\mu[$ ($\gamma_1 < 0$).

Cautionary Note.

Distribution systems like Pearson's are in general very flexible as they allow characterizing a broad range of combinations of moments. However, they require the estimation of several PDFs, and may face stability problems at the transitions between the different types of distributions (which may occur, e.g., by increasing the number of side-channel samples). Hence, in these cases, it is preferable to rely on a single distribution.

6.3.3 Shifted Generalized Lognormal

In [Low13], Low introduced the SGL distribution. It can be parameterized with the first four moments and covers a large interval of possible combinations of skewness and kurtosis. Both of these properties are desirable in side-channel evaluations, and therefore this distribution can be an interesting alternative to the Pearson's distribution system. The realm covered by the SGL is vast and we found it to be sufficient for all our practical experiments. This is illustrated by the plot of the distribution's coverage given in Figure 6.3.4 (which is similar to the aforementioned one given for Pearson's distribution system).

Concretely, the PDF of the SGL is given by:

$$F(x) = \frac{1}{2\lambda_3^{1/\lambda_3}\lambda_4\Gamma(1+1/\lambda_3)(x-\lambda_1)} e^{-\frac{1}{\lambda_3\lambda_4^{\lambda_3}}\left|ln\left(\frac{x-\lambda_1}{\lambda_2}\right)\right|^{\lambda_3}}, \tag{6.6}$$

for $\lambda_1 < x < \infty$, where λ_1, λ_2, λ_3, and λ_4 are the distribution parameters and $\Gamma(.)$ denotes the gamma function. These parameters can be estimated using the first four moments. For conciseness, we only give a brief overview of the resulting estimation problem and refer the interested readers to [Low13].

At the first step, we introduce a new variable Y defined as $Y = \frac{(X-\lambda_1)}{\lambda_2}$. The raw moments of Y can be computed using (λ_3, λ_4) as:

$$\mathsf{E}(Y^k) = \frac{1}{\Gamma(1/\lambda_3)} \sum_{i=0}^{\infty} \frac{(k\lambda_4)^{2i}}{(2i)!} \lambda_3^{2i/\lambda_3} \Gamma\left(\frac{2i+1}{\lambda_3}\right). \tag{6.7}$$

From these raw moments, the mean μ_Y, variance σ_Y^2, skewness γ_Y, and kurtosis β_Y of Y can be derived from the definitions given in Appendix 12. Given the actual mean μ_X, variance σ_X^2, skewness γ_X, and kurtosis β_X of X, we strive to find a pair (λ_3, λ_4) such that $(\gamma_Y, \beta_Y) = (\gamma_X, \beta_X)$. In [Low13] it is suggested to use Newton's method to approximate a vector $\mathbf{u} = [\lambda_4 \lambda_3]^T$ with:

$$\mathbf{G}(\mathbf{u}) = \begin{bmatrix} \gamma_Y(\mathbf{u}) - \gamma_X \\ \beta_Y(\mathbf{u}) - \beta_X \end{bmatrix} = 0. \tag{6.8}$$

In each iteration, the vector $\mathbf{u}$ is updated using the relation:

$$\mathbf{u}_{j+1} = \mathbf{u}_j - \mathbf{J}^{-1}(\mathbf{u}_j)\mathbf{G}(\mathbf{u}_j), \tag{6.9}$$

where $\mathbf{J}(.)$ is the Jacobian matrix defined as:

$$\mathbf{J}(\mathbf{u}_j) = \begin{bmatrix} \frac{\partial\gamma_Y(\mathbf{u}_j)}{\partial\lambda_4} & \frac{\partial\gamma_Y(\mathbf{u}_j)}{\partial\lambda_3} \\ \frac{\partial\beta_Y(\mathbf{u}_j)}{\partial\lambda_4} & \frac{\partial\beta_Y(\mathbf{u}_j)}{\partial\lambda_3} \end{bmatrix}. \tag{6.10}$$

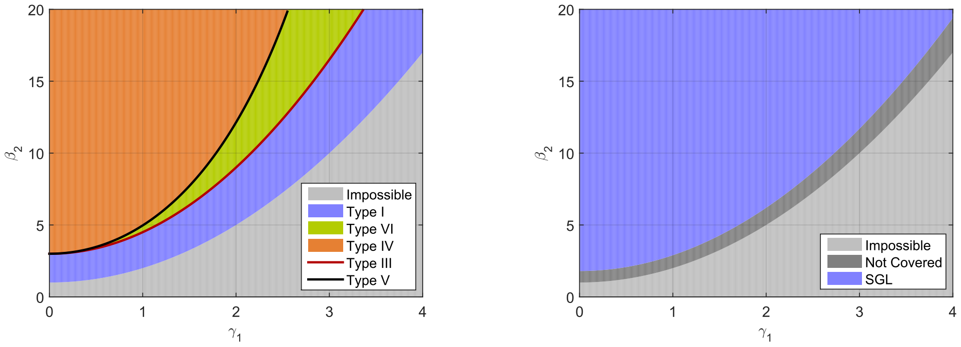
Once λ_3 and λ_4 are fixed, the other parameters can easily be computed by:

$$\lambda_2 = \frac{\sigma_X}{\sigma_Y}, \qquad \lambda_1 = \mu_X - \lambda_2\mu_Y. \tag{6.11}$$

Similar to the EMG, the SGL only considers positive non-zero skewness and needs to be mirrored for a negative skewness. Besides, and compared to the EMG and Pearson's system, this procedure has a higher computational complexity which can become significant if a large number of PDFs have to be estimated. For example, this can be the case for non-profiled attacks such as MIA that require computing PDFs for every possible subkey candidate. Indeed, our practical experiments employing SGL (presented in Section 6.5) required significant more time compared to the other considered estimators but remained tractable.

6.3.4 Coverage of Pearson and SGL

In Figure 6.2(a), the coverage for the different types of Pearson distributions is illustrated. Type I is limited by the impossible region ($\beta_2 \leq \gamma^2 + 1$). Type III covers the border between type I and type VI (i.e., $2\beta_2 = 3\gamma^2 + 6$). Similarly, the border between type VI and type IV is covered by type V ($\gamma_1^2(\beta_2 + 3)^2 = 4(4\beta_2 - 3\gamma_1^2)(2\beta_2 - 3\gamma_1^2 - 6)$). Note that we did not consider these two border cases (type III and type V) in our experiments. Figure 6.2(b) shows a similar coverage area for SGL distributions. In both cases, the non-covered realm of these PDF estimators is marked in gray to allow straight comparisons.



(a) Pearson's distribution system.

(b) SGL.

Figure 6.2: Plane of existence of the different distributions.

6.3.5 Computational Complexity

The presented parametric methods have all different PDFs with different computation complexities. For SGL, the computation of the parameters from the first four moments takes considerably longer than for all other discussed distributions. To present some intuitions on the run time of the different PDFs, we performed experiments using 100 randomly generated sets of moments and run each PDF[3] 100 times for each of these sets. Then we computed the average over all 10000 executions of each PDF. The Gaussian distribution is used as a reference value and has an average of 0.0034 s on an Intel i5-4200M CPU. The averages increase with the number of moments considered in the distribution: 0.0082 s (EMG), 0.029 s (Pearson), 1.70 s (SGL).

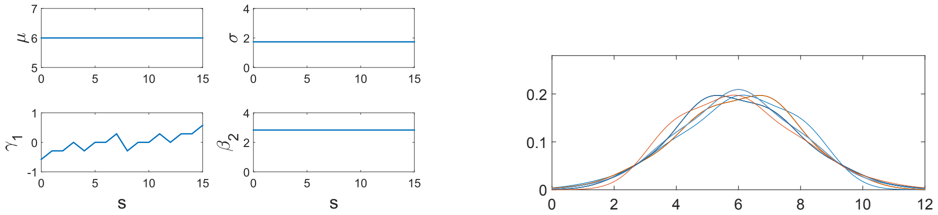
6.4 Simulated Experiments

In order to better understand the interest of the tools proposed in Section 6.3 in the context of side-channel analysis, we present a couple of simulated experiments. In the following we use mathematically-generated leakages derived from:

$$l = \mathrm{HW}(s \oplus c_1 \oplus c_2) + \mathrm{HW}(c_1) + \mathrm{HW}(c_2), \tag{6.12}$$

---

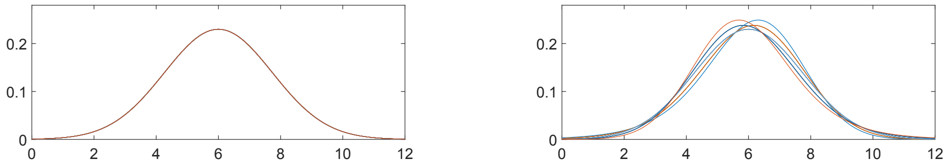
[3] We implemented three distributions in MATLAB and used the publicly available pearspdf [Bra].



(a) Estimated moments.

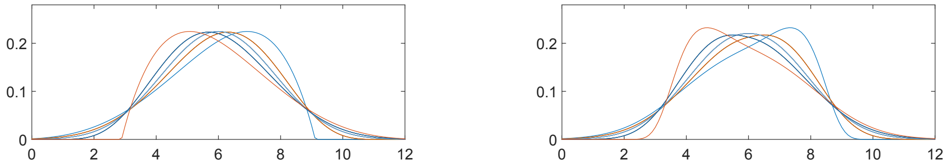
(b) Kernel-based PDF.

Figure 6.3: The estimated moments for each possible $s \in \{0,1\}^4$ (a) and kernel-estimated PDFs (b) for mathematically-generated leakages corresponding to a 2nd-order masking.



(a) Gaussian.

(b) EMG.



(c) Pearson.

(d) SGL.

Figure 6.4: The estimated PDFs for mathematically-generated leakages corresponding to a 2nd-order masking, obtained with various parametric tools from Sections 6.2 and 6.3.

where HW(.) denotes the Hamming weight function, s a sensitive (secret) 4-bit variable, and c_1 and c_2 uniformly distributed random masks in $\{0,1\}^4$. Note that this example is related to any nibble-oriented cipher, e.g., PRESENT [BKL+07], and the basic evaluation procedure presented in this paper does not change for larger bit sizes. The only adjustment is the number of possible different classifications, i.e., 2^n instead of 2^4 for n-bit variables. In this simulation it is supposed that the target is a hardware design where the shares are processed at the same time. This scenario essentially emulates a second-order Boolean masking scheme, where we only focus on the encoding of a single variable s in a noise-free situation. In this context, the first and second moments of the leakage distribution are expected to be independent of s. For each $s \in \{0,1\}^4$, we estimate the PDF using both non-parametric (kernels) and parametric (Gaussian, EMG, Pearson, SGL) tools. The first four moments for each s, plotted in Figure 6.3(a), reveal that there is indeed no dependency between s and the first two moments (i.e., they remain constant for all s). Hence, the only way that s can be distinguished is by observing the third moment. Since kernel-based density estimation considers all possible moments, it can be used to distinguish s as shown in Figure 6.3(b).

By contrast, the third moment is not used to parameterize the Gaussian distribution and thus each s results in the same distribution in this case (as per Figure 6.4(a)). This example shows why Gaussian density estimation cannot be used to analyze the leakages that reside in an order higher than two. Eventually, our newly proposed estimators consider moments up to the fourth one, and therefore they can be used to quantify the information leakage of our simulated masking experiment (this can be seen in the remaining part of Figure 6.4).

6.5 Practical Case Studies

To examine the application and efficiency of the above-mentioned solutions, we consider a threshold implementation of the PRESENT cipher [BKL$^+$07] on an FPGA platform. More precisely, the target design is the *Profile 2* presented in [PMK$^+$11] that follows a serialized architecture, i.e., using one instance of the Sbox for the whole SLayer. Such a masked hardware implementation has been selected for the practical investigations due to its second- and third-order univariate leakages which allow us to examine our proposed tools. If we would have no leakage at order three and higher, examining the difference between our tools and Gaussian would not be possible.

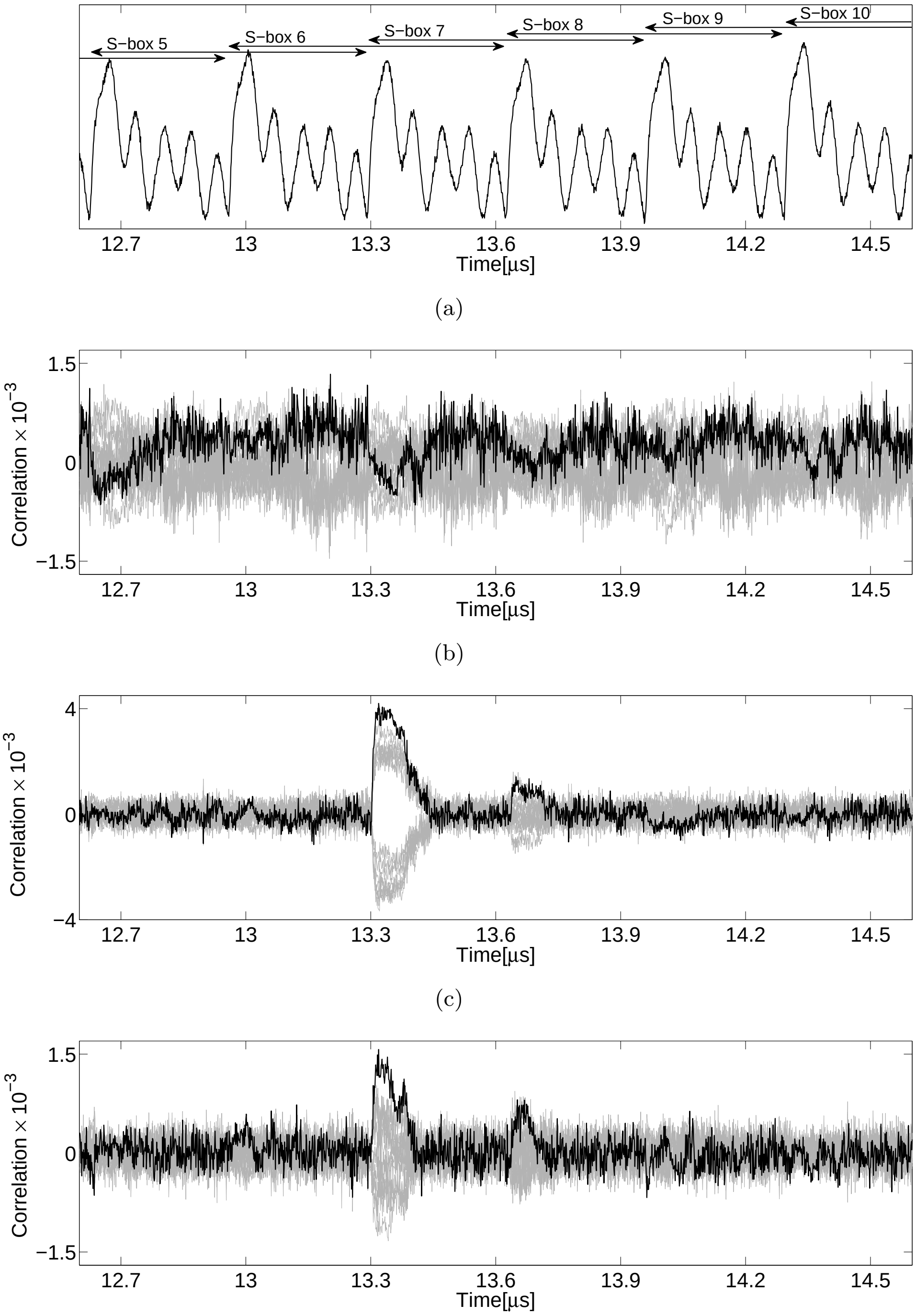
In the target implementation, the data state is represented by $d = 3$ Boolean shares, and the SLayer is based on the 2-stage masked Sbox described in [NRS11]. In other words, each Sbox on a 4-bit data is implemented in a pipeline fashion and needs two clock cycles to be computed. For more details on the design architecture we refer the interested reader to [MS14] and [PMK$^+$11].

The leakage traces are collected from a Xilinx Virtex-II Pro FPGA embedded on SASEBO [Sak]. The sampling rate was set to 1 GS/s and the target FPGA clock was driven at a frequency of 3 MHz. Figure 6.5(a) shows an exemplary trace covering six clock cycles with respect to the full computation of 5 Sboxes on 5 key-whitened plaintext nibbles.

We collected $100,000,000$ traces to be used in our experiments. During the measurements, the PRNG that provides random data (masks) for the sharing of the plaintext was kept active. We also examined and confirmed the uniform distribution of the masks.

A former analysis of MCP-DPA by Moradi and Standaert in [MS14] on the same implementation revealed that the first pipeline stage of the target Sbox exhibits the most informative leakages. The result of such an analysis is given in the lower part of Figure 6.5 for completeness. It confirms that no first-order leakage can be exploited from this implementation, whereas the second and third moments are indeed informative. It also suggests that second-order leakages are more informative than third ones. By contrast, and as exhaustively discussed in the introduction, two important questions remain open. First, can we quantify the informativeness of the different moments on a (somewhat) more formal basis? Second, and given that more than a single moment provides information, can we design an attack that jointly exploits these moments? (which is in contrast with MCP-DPA that only exploits moments one by one).

Both questions can be answered in the affirmative by the following discussion. In order to make our results comparable with [MS14], we focus on the same parts of the leakage traces. Namely, we analyze the most informative clock cycle in the Sbox execution that corresponds to samples between $13.3\,\mu$s and $13.6\,\mu$s in Figure 6.5(a). Based on this case study, we show that the newly introduced PDF estimation tools are powerful ingredients for the information theoretic analysis of a threshold implementation. First, they are able to extract an amount of information from the traces comparable to a kernel density estimation. Second, they are useful to estimate the informativeness of each moment, and to perform attacks based on the



(a)

(b)

(c)

(d)

Figure 6.5: (a) sample trace. (b) first-order, (c) second-order, and (d) third-order MCP-DPA results for different time samples in the leakage traces (taken from [MS14]).

best combination of moments carrying significant information. Eventually, they can naturally and efficiently be embedded in PDF-based non-profiled attacks such as MIA.

6.5.1 Profiled Evaluations and Attacks

First, we examine the information leakage of the target device using an information theoretic approach. The idea to use MI as an evaluation metric was introduced in [SMY09]. It was later refined in [RSV⁺11] to incorporate the fact that the leakage distribution is only estimated, which can potentially bias the estimation of the MI. The so-called Perceived Information (PI) is used to reflect this bias and can be computed as:

$$\hat{PI}(S;L) = H[S] - \sum_{s\in\mathcal{S}} Pr[s] \sum_{l\in\mathcal{L}} Pr_{\mathsf{chip}}[l|s] \cdot log_2 \hat{Pr}_{\mathsf{model}}[s|l], \tag{6.13}$$

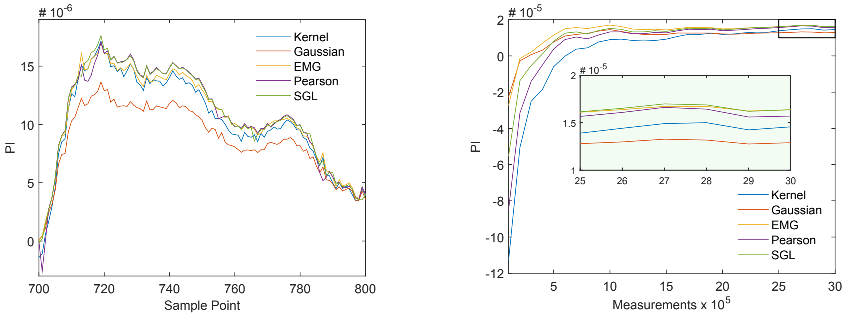
where Pr_{chip} denotes the chip's true distribution (which is unknown but can be sampled) and $\hat{Pr}_{\mathsf{model}}$ refers to the adversary's estimated model (for which we have an analytical formula). Computing the PI essentially requires an estimated $\hat{Pr}_{\mathsf{model}}$, which is exactly what our PDF estimation tools provide. In our experiments, we followed the procedure presented in [DSV14] for computing this metric. In particular, we used 10-fold cross-validation and report the mean of the resulting PI estimates. We start by looking at the information extracted using all the moments enabled by each PDF estimation tool. We then analyze (subsets of) these moments separately.

Combined Moments.

In order to compare our proposed solutions (EMG, Pearson, SGL) with the established ones (kernels, Gaussian), we first compute the PI using all the covered moments. We estimate $\hat{Pr}_{\mathsf{model}}$ using the different estimators and compare the results. As previously mentioned, this experiment only covers 100 sample points corresponding to the power peak of the targeted clock cycle, i.e., between 13.3 μs and 13.4 μs in Figure 6.5(a). The 100,000,000 traces are divided into 10 sets. For each of the 10 runs we use one of these 10 sets (each with 10,000,000 measurements) as samples of the chip's true distributions, and the remaining 9 sets (90,000,000 measurements) to estimate the model distribution ($\hat{Pr}_{\mathsf{model}}$). Figure 6.6(a) contains the results.

At the first glance, it can be observed that the achieved PI using the Gaussian distribution to estimate $\hat{Pr}_{\mathsf{model}}$ is the lowest. This implies that not all available information is contained in the first two moments (that are the only ones captured by a Gaussian distribution). More interestingly, kernel-based density estimation is non-parametric and therefore is expected to provide the highest PI if its bandwidth is well adapted and enough samples are available. Yet, we observe that this is not exactly the case in our experiments. As depicted in Figure 6.6(b) (where we focus on the most informative sample 719), this is most likely due to an estimation issue (i.e., a lack of samples). As expected, the non-parametric kernel density estimation is the slowest to converge in this case. This suggests an interesting feature of our new parametric tools. Namely, whereas Gaussian estimation is very fast but limited to the exploitation of two moments (hence leads to less efficient attacks, as will be discussed next), EMG-, Pearson- and SGL-based estimations combine a faster convergence than kernels with a similar informativeness.

Summarizing, we can conclude that PDFs covering the right combination of moments lead to the best trade-off between a fast convergence towards a well estimated model, and a well-



(a) 100 sample points of the power peak.

(b) At sample point 719.

Figure 6.6: Kernel-, Gaussian-, EMG-, Pearson- and SGL-based PI estimation with all covered moments (a) using 100,000,000 meas., (b) over the number of meas.

informative model once properly estimated (i.e., a model for which the PI should be close to the MI [DSV14]). By contrast, the previous results do not allow deducing about the relative informativeness of each moment (which could be used to further speed up the model estimation and attacks), which motivates the following analysis.

Separate Moments.

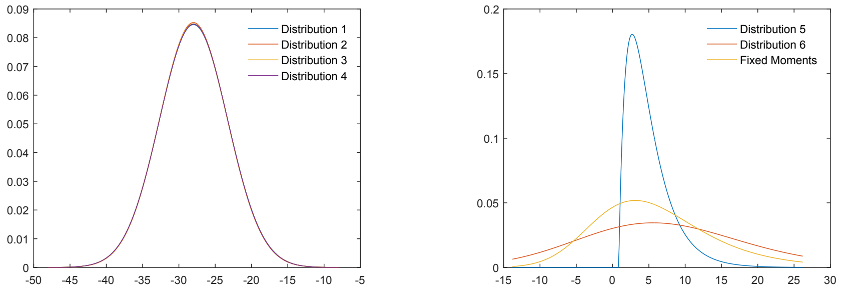
An interesting property of the parametric estimators is the ability to consider only selected moments instead of trying to characterize any possible moment (as in non-parametric estimations). Using the Gaussian distribution as an example, we can compute the information contained exclusively in the first two moments, as this distribution only considers the mean and variance. Similarly, it is also possible to compute the PI for the first three moments (with EMG distributions) and the first four moments (with Pearson's distribution system and SGL distributions). In the following, we present an approach that enables us to compute the PI both for each moment taken separately and for any combination of those.

For this purpose, and taking the case where we focus on a single moment, we simply have to set all but one of the moments to a fixed value. For example, suppose that we want to consider the information contained in the first moments of a Gaussian distribution only. We achieve this by considering a Gaussian model where the means are estimated as in the previous section, but the variances are set to a fixed value, which essentially removes any secret-dependent information they could carry from the templates through the second moments. Since changing the variances affects the shape of the distributions, the fixed value can be chosen as the average of the variances (over the 16 templates) to minimize the distance between the original distributions and the ones with a fixed variance[4]. A similar technique actually works for any of our parametric estimators, and for any (combination of) moments.

When we set specific higher-order moments (as in our approach) to specific values, we actually fix the width of the distributions (i.e., variance), or their right-left tendency (i.e., skewness) or

---

[4] Instead, one can also consider the variance of whole trace set. Here we need only a fixed value which is not too different from the variance of each template. Such an approach is not valid in case of Gaussian mixtures as stated in Section 6.2.



(a) At point 719 (cf. Table 6.1).

(b) Simulations (cf. Table 6.2).

Figure 6.7: The PDFs of the six distributions from Table 6.1 and 6.2.

Table 6.1: The first four statistical moments of four distributions at sample point 719.

	Dist. 1	*Dist. 2*	*Dist. 3*	*Dist. 4*	*Average*
Mean	-27.9734310	-27.9811494	-27.9827913	-27.9782609	-27.9789082
Variance	22.3624316	21.9979663	22.2165081	22.2660171	22.2107308
Skewness	0.0075083	0.0053184	0.0131009	-0.0000767	0.0064627
Kurtosis	3.0177549	3.0202503	3.0219293	3.0183596	3.0195735

their sharpness (i.e., kurtosis). Hence, information sitting in the corresponding moments does not contribute in the information-theoretic-based evaluation, e.g., mutual information. We like to emphasize that the estimated higher-order moments in real side-channel measurements (categorized, for example, based on the processed data) are very slightly different. Consider for example the PDFs of four exemplary distributions shown in Figure 6.7(a), taken from the most leaking point of the measurements of our case study (see Figure 6.6(a)). The first four moments of each distribution are given in Table 6.1. All moments of the four distributions are very similar to each other, e.g., the skewness of all these four distributions is only slightly different. Hence, fixing the skewness of all of them to a specific value (e.g., the average of all skewnesses given by 0.0064627) does not significantly change the shape of the distributions.

Here we consider four different cases:

(1) All moments except the first are fixed to their average (evaluation through means).

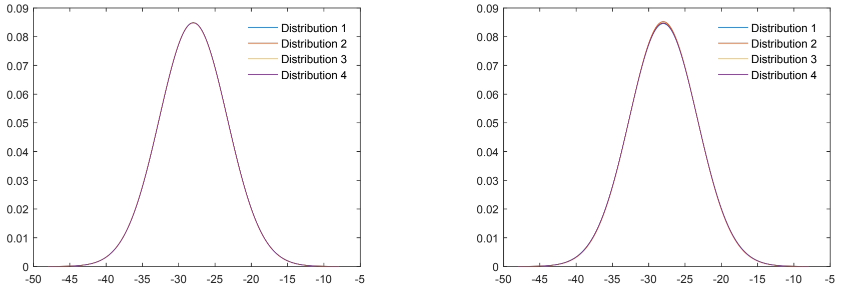
(2) All moments except the second are fixed to their average (evaluation through variances).

(3) All moments except the third are fixed to their average (evaluation through skewnesses)

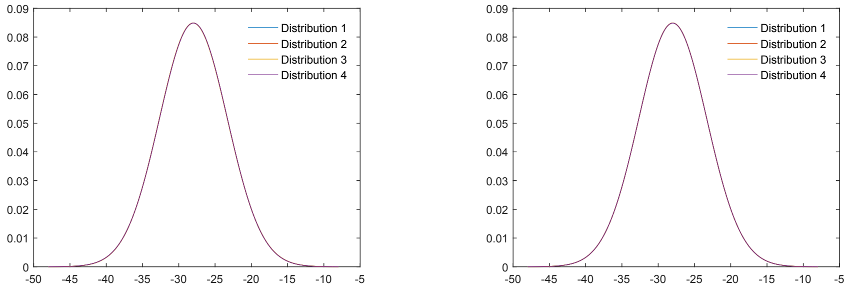
(4) All moments except the fourth are fixed to their average (evaluation through kurtoses).

For each case, the shape of the resulting distributions is very close to the original shape in Figure 6.7(a). The PDFs of the modified distributions for each case is provided in Figure 6.8.

It should be noted that in case of simulated data with significantly different moments for each distribution the resulting shapes of each distribution would be also dramatically different to each other. Therefore, in this case, setting the corresponding moments to a fixed (average)



(a) Evaluation through means. (b) Evaluation through variances.



(c) Evaluation through skewnesses. (d) Evaluation through kurtoses.

Figure 6.8: The estimated PDFs of the four distributions from Table 6.1 with partly fixed moments according to the four evaluations cases.

value does not make the distributions to roughly follow the same shape. If such a huge difference between the moments of the (categorized) distributions exists in practice by any (rare) chance, the corresponding implementation is significantly vulnerable to certain attacks. Obviously, this makes the necessity of performing per-moment evaluations questionable. As an example, we show in Figure 6.7(b) two simulated distributions formed by the moments from Table 6.2. It is obvious that the shape of the distribution with fixed moments is considerably different from the original two distributions. In this case, a per-moment approach would not be easily possible with an information-theoretic evaluation tool.

We analyze this moment-based investigation based on the same case study as for the previous information theoretic analysis. Hence, we repeat the previous experiments (of Figure 6.6(a)) with the same parametric estimators (Gaussian, EMG, Pearson, SGL), but this time we consider each possible moment separately. The results are depicted in Figure 6.9 where the PI curves are categorized based on the employed estimator. Each part of the figure contains the PI curves obtained for each moment separately. For example, in Figure 6.9(a) the curve labeled *Gaussian (1st)* shows the PI achieved for the first moments (and the curve *Gaussian (2nd)* depicts the same for the second moments, etc.). Further, we included the PI curve of the combined moments (taken from Figure 6.6(a)) and the sum of the PI curves of the separate moments (e.g., *Gaussian Sum* as the sum of the PI curves of *Gaussian (1st)* and *Gaussian (2nd)*).

Table 6.2: The first four statistical moments of two simulated distributions.

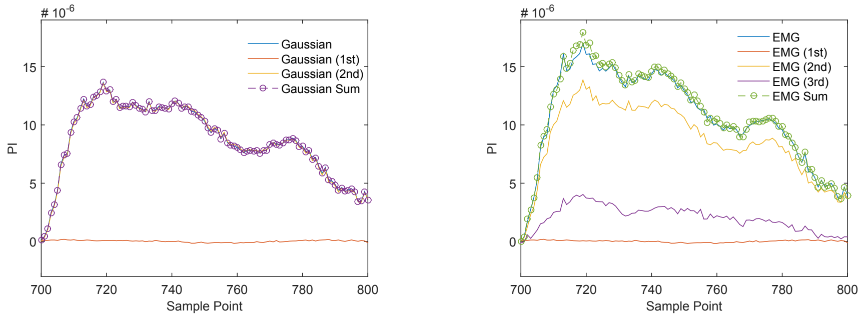
	Dist. 5	*Dist. 6*	*Average*
Mean	4.9997939	7.400773	6.2002834
Variance	10.0032941	149.017440	79.5103671
Skewness	1.7063003	0.377136	1.0417184
Kurtosis	7.8417563	3.648649	5.7452030

As expected, the first moment does not contain any exploitable information as the implementation is first-order secure. It is also noticeable that the chosen estimator does not affect the PI for the first moment. The second moment leads to the highest PI, and therefore is the most informative moment. As similarly indicated by MCP-DPA, the third moment is informative but not as much as the second one. Furthermore, using two estimators (Pearson, SGL) that also cover the fourth moment, we are not able to detect any significant information leakage in the fourth moment. Therefore, a combination of the second and third moments should suffice to capture most of the available information in the underlying measurements.

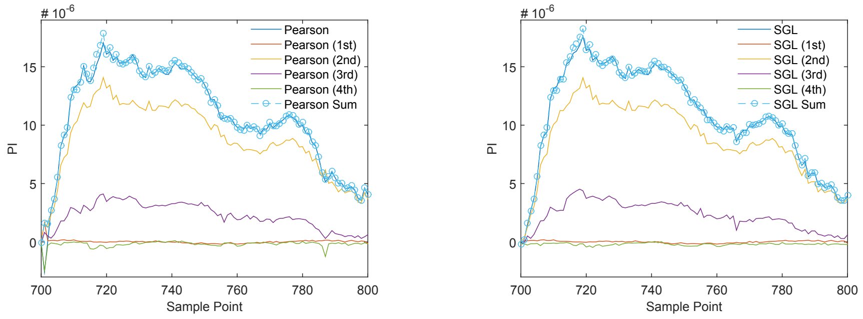
Most interestingly, we observe that the sum of the PI values obtained for the separate moments is actually close to the PI estimated with the combined moments. Although informal, this observation is particularly interesting in view of the recent results by Duc *et al.* in [DFS15] where the PI values are connected with the success rate of a (worst-case) template attack using the same model. Indeed, since the sum of the PI values obtained per moment is essentially the same as the PI value obtained with the non-parametric kernel method, it implies that *in our case study*, the separation between moments did not lead to any significant information loss. This suggests that a (simple and intuitive) moment-based side-channel evaluation could be well-founded, at least in certain contexts that would be worth formalizing. And very concretely, it also means that an attack exploiting our two informative (i.e., second and third) moments will be close to optimal in our case.

Profiled attacks.

The results in [DFS15] prove that (under sufficiently noisy leakages) the success rate of a profiled template attack is inversely proportional to the PI value estimated with the same model. In view of the previous discussions, it means that our proposed estimation tools (EMG, Pearson, SGL) should lead to more effective profiled attacks than their counterparts with Gaussian estimation (because of modeling errors) and kernels (because of assumption errors). Furthermore, the attacks exploiting the second moment should lead to a higher success rate than attacks exploiting the other three moments. Eventually, the best attack should exploit the combination of second and third moments. For completeness, we ran experiments to confirm these expectations. We built univariate templates (for the most informative sample point 719) from 90,000,000 measurements and, for each given number of measurements, repeated an attack 1000 times for different measurements (excluding those used for profiling) to compute a subkey recovery success rate. The results of this last experiment are depicted in Figure 6.10 and are well in line with theoretical predictions. In this respect, the most interesting curves are the ones corresponding to the combination of second and third moments, since they correspond to the



(a) Gaussian. (b) EMG.



(c) Pearson. (d) SGL.

Figure 6.9: PI estimates for the separate moments.

best trade-off between model complexity and attack efficiency, and could not have been reached with existing side-channel evaluation tools. Additional curves are provided in Figure 6.12(a) which includes attacks exploiting kernel-based models that are as efficient, but as mentioned earlier, more expensive to estimate.

6.5.2 Non-Profiled Attacks

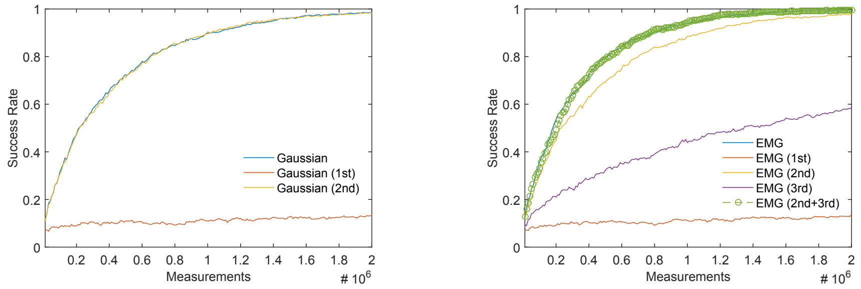
In addition, we briefly discuss the application of our solutions in the non-profiled attack setting. For this purpose, we consider a univariate MIA, which is the standard representative for non-profiled attacks exploiting PDF estimation. As usual in this context, we cannot directly use a generic (i.e., identity) power model, since it would not be able to extract any key-dependent information [WOS14] if the first encryption round is targeted[5]. Further, MIA needs a non-bijective model to be effective. After examining many models,[6] we selected the three most significant bits of the Sbox output as the best alternative.

Using each density estimator (with various combinations of moments), we further ran 1000 MIA experiments for each given number of traces, and computed the guessing entropy as defined in [SMY09]. The reason for not using the success rate again is that the convergence of the attacks is not guaranteed in this case (and actually not all the attacks converged). The results depicted

---

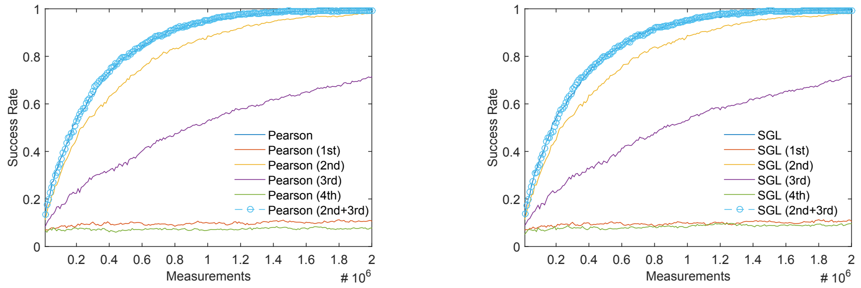
[5] Such an identity model is applicable to e.g., the Sbox output of the second encryption round [RGV14].

[6] Including HW, any single bit, pair and triple of bits of the Sbox output.



(a) Gaussian.

(b) EMG.



(c) Pearson.

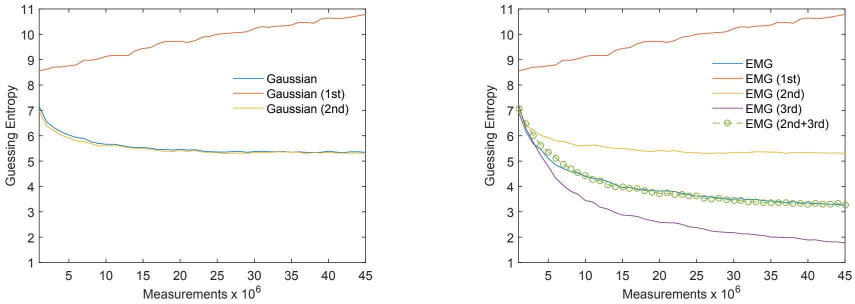
(d) SGL.

Figure 6.10: Success rate of several univariate template attacks exploiting separate and combined moments, for the most informative sample point 719 in our traces.

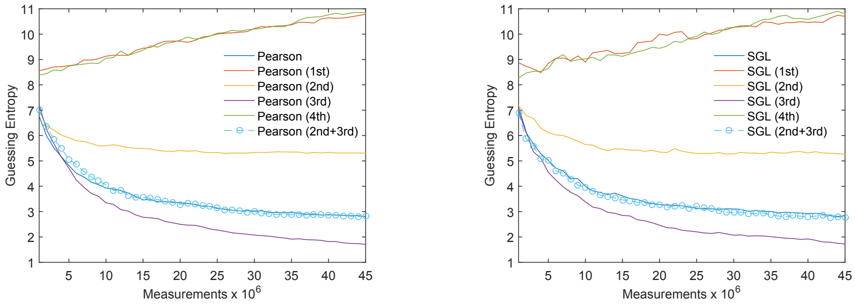
in Figure 6.11 indicate that the estimators that capture more of the available moments generally perform better. Yet, the most interesting (and somewhat surprising) fact is that the most useful moment is now the third one rather than the second one. A similarly interesting observation is that the best attack is not the one combining all moments. This is not contradictory with the previous analysis, since such non-profiled attacks naturally deviate from the worst case predictions based on the profiled PI values. Indeed, in the case of MIA, the estimation of the model parameters is performed “on-the-fly”, which implies that the best option is not to characterize the leakage the most carefully, but to reach a sufficiently precise estimation sufficiently quickly. Besides, our experiments also indicate that (non-profiled) models that are useful for certain moments (and as a matter of fact, certain time samples as well) may not be as good for others. This somehow joins the conclusions in [SVO$^+$10] regarding the difficulty to interpret the result of non-profiled side-channel attacks in the context of masking.

6.5.3 Selection of Tools

We have discussed multiple parametric tools, each with its own advantages and disadvantages. Compared to the traditional non-parametric tools, they offer a higher flexibility and convergence. Therefore, they should be preferred if the number of samples is too small or a special case (e.g., only two moments) should be evaluated. The PDF of EMG can be computed very efficiently compared SGL and Pearson. However, it considers only the first three moments instead of four.



(a) Gaussian (b) EMG

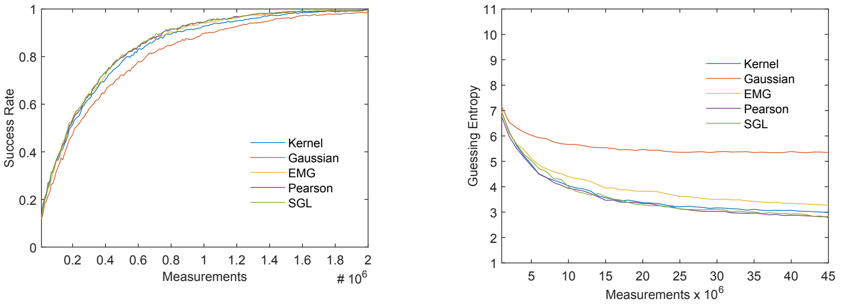


(c) Pearson (d) SGL

Figure 6.11: Guessing entropy for MIA based on different estimation tools (at sample 719).

The Pearson distribution system includes the kurtosis and its PDF is still relatively efficient compared to SGL. Nevertheless, it is made up of multiple different distributions which can be problematic in certain cases as pointed out in Section 6.3.2. Therefore, in scenarios where the computation time of the PDF can be ignored and the leakages are covered by SGL, it is the preferable tool.

However, the computation time is often a limiting factor and it can be significantly reduced in certain cases by choosing a more limited distribution which is still sufficient to capture all relevant leakage. If the type of implementation and leakage is known, this choice is easily possible. Gaussian (resp. EMG) is the preferred choice for leakage which is exclusive in the first two (resp. three) moments due to its very efficient PDF. Leakage in the fourth moment can be also efficiently captured with the Pearson distribution system, assuming that the aforementioned problems do not arise. If the type of masked implementation, i.e., the order of masking, is unknown, then this choice of distribution cannot be that easily made. SGL is then the best approach, if the distribution is inside the plane of existence of SGL. For the separate moments method, it is still possible to reduce the computation time by using some of the other distributions (Gaussian, EMG) for the moments of lower order.



(a) Profiled Template Attack.

(b) Non-profiled MIA Attack.

Figure 6.12: Additional (profiled and non-profiled) attacks with kernel density estimation and comparison with other attacks exploiting all the moments at time sample 719.

6.6 Conclusion and Future Work

This chapter introduced a variety of PDF estimation tools to improve the evaluation of leaking devices, both in the profiled and non-profiled settings. Their main interest is their flexibility: our proposals can indeed capture information lying in different moments of the leakage PDF. As a result, we can easily analyze masked implementations and extract useful feedback to hardware designers, i.e. in terms of how much information is lying in every moment and how to combine it. This brings a concrete and more founded counterpart to the recent evaluations of implementations with non-independent leakages in [DFS15], where this quantity of information "per moment" is required. More generally, our findings provide efficient trade-offs between the cost of profiling and the efficiency of the resulting attacks, since they allow adversaries and evaluators to build models that are tailored to their implementations.

These results naturally open various interesting research challenges for future work. As mentioned in the introduction, combining an analysis of moments as in this work with simplifying approaches to leakage modeling (e.g. based on linear regression) would be even more convenient to evaluators. Besides, investigating the "summing rule" of Section 6.5.1 more formally is certainly worth further efforts as well. Eventually, our current tools are limited to univariate leakages. While this was sufficient to analyze our hardware case study, it naturally suggests the extension to multivariate case studies as yet another important question. This is especially interesting given that even hardware designs with univariate d-order security may include a multivariate vulnerability for which less than d points are combined [RBN+15]. A starting point for this purpose would be to exploit some popular "combining" functions from the side-channel literature (which would allow us to exploit our univariate tools directly). Furthermore, an additional limiting factor for some distributions is the computation complexity of the PDF. It is worth investigating how it can be improved, e.g., by utilizing some results of the previous two chapters.

Part III

Advanced Countermeasures against Physical Attacks

Chapter 7

Background: Countermeasures

In this chapter, we give background information relevant to the research contribution of this part. To this end, the concept of threshold implementation is introduced which enables the secure realization of masking in hardware. The descriptions are partly based on [SMG15b, BGG⁺16a, SMG16b].

Contents of this Chapter

7.1 Threshold Implementations 87

7.1 Threshold Implementations

The first attempts to realize Boolean masking in hardware were unsuccessful, mainly due to glitches [MPO05, MME10]. Combinatorial circuits which receive both the mask and the masked data, i.e., secret sharing with 2 shares, most likely exhibit first-order leakage. In response, the concept of Threshold Implementation (TI) [NRR06], a mixture between Boolean masking and multi-party computation, has been specifically developed for hardware platforms to maintain security properties even in the presence of glitches. Further publications have extended the concept to higher-order security [BGN⁺14b] and examined the relation of threshold implementation to other masking schemes [RBN⁺15]. The TI concept has been applied to many algorithms including PRESENT [PMK⁺11], AES [MPL⁺11, BGN⁺14a, CBR⁺15], KATAN [BGN⁺14b, MW15], Keccak [BDN⁺13], Simon [STE15], PRINCE and Midori [MS16d], and all 4-bit Sboxes [BNN⁺15]. We also rely on the concept of threshold implementation for our countermeasures which are presented in the subsequent chapters. To this end, we briefly recall all aspects of threshold implementations which are relevant to our work.

TI is a masking scheme which splits any secret data x into several shares x^i, using a simple Boolean secret-sharing scheme with the relation

$$x = \bigoplus_i x^i. \tag{7.1}$$

A target function F with $y = F(x)$ is transformed into multiple component functions $f^j(\dots)$ which compute the output shares $(y^1, \dots, y^{s_{out}})$ given a subset of the input shares $(x^1, \dots, x^{s_{in}})$. To ensure the desired side-channel resistance, these component functions need to comply to three basic properties [BGN⁺14b].

Correctness. For every $(x^1, \ldots, x^{s_{in}})$ the masked functions should provide the correct output in the shared form $(y^1, \ldots, y^{s_{out}})$ with $\bigoplus_{i=1}^{s_{out}} y^i = y = F(x)$ and $s_{out} \geq s_{in}$.

(d-order) Non-completeness. Each component function computes on a specific subset of the input shares $(x^1, \ldots, x^{s_{in}})$. A shared function is d-order non-complete, if any combination of d of these input share subsets is independent of at least one of the input shares.

Uniformity. Suppose that for a certain input x all possible sharings $\left\{({}_1x^1, \ldots, {}_1x^{s_{in}}), \ldots, ({}_px^1, \ldots, {}_px^{s_{in}})\right\}$ are given to the TI Sbox. The tuple $(f^1(\ldots), \ldots, f^{s_{out}}(\ldots))$ should be drawn uniformly from the set $\left\{({}_1y^1, \ldots, {}_1y^{s_{out}}), \ldots, ({}_qy^1, \ldots, {}_qy^{s_{out}})\right\}$ as all possible sharings of $y = F(x)$.

Furthermore, based on the algebraic degree t of the targeted non-linear function as well as the desired order of security d, the minimum number of input shares s_{in} and the minimum number of output shares s_{out} are defined as

$$s_{in} = t \times d + 1 , \qquad\qquad s_{out} = \binom{s_{in}}{t}.$$

An important point is that the output of the component functions must be stored in dedicated registers to avoid the propagation of glitches. If this is ensured and the shared functions complies with the aforementioned properties, the resulting hardware design will provide the desired degree of side-channel resistance. However, to achieve complete higher-order resistance it is necessary to perform some additional design steps. Firstly, there is an issue related to the uniformity of the TI functions of security order $d > 1$. In such a case, the number of output shares s_{out} is usually higher than the number of input shares s_{in}; hence uniformity cannot be achieved. Therefore, some registered output shares should be combined to reduce the number of output shares to s_{in} at most. After such a combination, the uniformity can be examined. Secondly, as noted in [RBN+15], this approach only provides univariate higher-order security and is still vulnerable to multivariate higher-order attacks. In Chapter 8, we practically verify this issue for our higher-order TI design. In [RBN+15], the authors propose to include additional randomness to solve this issue and achieve univariate and multivariate higher-order security.

The transformation of an arbitrary function F into its shared representation can lead to a significant area overhead which mostly depends on the algebraic degree of the function. A higher degree requires more input shares which in turn require more hardware resources. Therefore, functions of a particular high degree are often decomposed into multiple functions with smaller degrees, e.g., functions F_1, F_2 with $y = F(X) = F_1(F_2(x))$. Since most block ciphers include elements with a high degree (i.e., Sboxes), extensive research has been done to efficiently decompose these functions. Especially, 4-bit Sboxes were the focus of a lot of publications including [BNN+15] in which the author identify 302 equivalence classes. However, finding a good decomposition for 8-bit Sboxes is still challenging due to the large search space. Therefore, all TI of the AES Sbox [MPL+11, BGN+14a, CBR+15] require additional randomness to achieve the uniformity property.

Recently, new schemes [CRB$^+$16, CFE16, GMK16] building on [RBN$^+$15] have been proposed which only require $d+1$ shares for d-order masking. The number of shares is therefore independent of the algebraic degree which enables masked design with only two shares and thus helps to decrease the required design area. However, the authors of [CFE16] hint that a two-share design might lead to less practical security due to being vulnerable to second-order attacks for a small number of measurements.

Chapter 8

Arithmetic Addition over Boolean Masking in Hardware

A common countermeasure to thwart side-channel analysis attacks is algorithmic masking. For this, algorithms that mix Boolean and arithmetic operations need to either apply two different masking schemes with secure conversions or use dedicated arithmetic units that can process Boolean-masked values. Several proposals have been published that can realize these approaches securely and efficiently in software. But to the best of our knowledge, no hardware design exists that fulfills relevant properties such as efficiency and security at the same time. In this chapter, we present two design strategies to realize a secure and efficient arithmetic adder for Boolean-masked values based on [SMG15b]. First, we introduce an architecture based on the ripple-carry adder that targets low-cost applications. The second architecture is based on a pipelined Kogge-Stone adder and targets high-performance applications. In particular, all our implementations adopt the threshold implementation approach to improve their resistance against SCA attacks even in the presence of glitches. We evaluated the security of our designs practically against SCA using a non-specific statistical t-test. Based on our analysis, we show that our constructions not only achieve resistance against first- and (univariate) second-order attacks but also require fewer random bits per operation compared to any existing software-based approach.

Contents of this Chapter

8.1 Introduction **91**
8.2 Background **93**
8.3 Implementation **94**
8.4 Analysis **99**
8.5 Conclusion and Future Work **104**

8.1 Introduction

In this work we focus on masking schemes developed to be applied at algorithmic level, e.g., Boolean and arithmetic masking, which need to be adjusted according to the underlying cryptographic algorithm [MOP07]. Note that nearly all proposed ciphers employ both logical and arithmetic operations.

As an example, ARX-based designs consist of three operations: integer addition, rotation, and XOR. Such constructions are the foundation for block ciphers (like FEAL [Miy90] or Threefish [FLS$^+$10]), stream ciphers (Salsa20 [Ber08b], ChaCha [Ber08a], HC-128 [Wu08]) and hash functions (BLAKE [AHMP08], Skein [FLS$^+$10]). There are further examples that also include a mixture of Boolean and arithmetic operations like the TEA family of block ciphers [WN94] and SHA-2 [NIS12]. To realize a masked implementation of these constructions, one option is to employ both Boolean and arithmetic masking schemes. Rotation and XOR operations can be protected by Boolean masking, while arithmetic masking is advantageous for the addition operations. However, the required conversions between both operations can also be the target of an SCA attacker and hence need to be implemented securely. In particular, many existing results discussing this method identified the conversion between arithmetic and Boolean masking as a major hurdle [BP10a, GBC$^+$08, BLV12].

8.1.1 Related Work

We now briefly highlight several works on the conversion between Boolean and arithmetic masking. The conversion techniques can be categorized into those which use *precomputation* [Deb12] and those *without precomputation* [Gou01]; however most of them were designed specifically for software platforms. Unfortunately, these constructions cannot be easily mapped to a dedicated hardware module without violating their claims on security. Roughly speaking, this is mainly due to critical glitches that occur inside masked circuits [MPO05]. To avoid this problem, every step would need to be separated by a register stage which would be detrimental to the performance.

We like to remark that a hardware design for such conversions has been proposed in [Gol07], but since both the mask and masked data are involved in the processes of the proposed techniques, such constructions are expected to still have first-order leakages (see [MME10]). Another problem is the transformation of conversion algorithms to higher orders. It has been shown in [CGV14] how to secure the conversions against higher-order attacks, but this feature comes with a prohibitive overhead for any cryptographic implementation.

Along the same lines, in order to avoid the conversions a technique to securely perform modular arithmetic addition on Boolean-masked operands has been introduced in [KRJ14]. However, this scheme has been developed to be used in software applications and cannot be easily applied on a hardware platform where performance is a key factor.

Recently, an approach was developed in [CGTV15] which uses the Kogge-Stone adder as a basis. But the conversion and masked addition requires more random bits compared to the solutions from [Gou01] and [KRJ14] and are only faster for larger bit sizes (i.e., 64 bits). Still, their focus lies on software applications which makes them inefficient in hardware.

8.1.2 Contribution

The target of this work is to design efficient hardware modules for modular addition of Boolean-masked operands. More precisely, our goal is to develop a similar technique such as [KRJ14] for a hardware platform. Since masked hardware designs face severe challenges due to glitches, we apply the concept of threshold implementations (TI) [NRS11] that can satisfy the security requirements even in the presence of glitches.

In this paper we consider two factors to design the aforementioned module: (a) *throughput* and (b) *SCA security order*. With respect to performance (i.e., throughput) we consider two designs to implement a 32-bit arithmetic adder that is required by many cryptographic algorithms:

(1) Ripple-Carry Adder (RCA) that requires 32 clock cycles to perform a complete addition,

(2) Kogge-Stone Adder (KSA) with 6 clock cycles latency and a fully pipelined architecture.

We present the first-order and (univariate) second-order secure threshold implementation of the two above mentioned designs. We show that our designs not only outperform the inefficient approaches of [CGV14] but also reduce the number of fresh random mask bits required for each addition. We also present practical SCA evaluations performed on a Spartan-6 FPGA to confirm the claimed security levels. To the best of our knowledge, our four proposed architectures are the only available hardware-dedicated solutions that are supported by security proofs as well as by practical investigations.

8.2 Background

In this section, we introduce the used notations and present the basic ideas behind our designs.

8.2.1 Notation

In the following sections all equations are bit-level operations. An n-bit integer operand a is represented as $(a_{n-1}a_{n-2}\cdots a_1a_0)$ where a_0 is the least significant bit. These integers are split up into shares of which the j-th share of a bit $a_{i\in\{0,\ldots,n-1\}}$ is denoted by a_i^j. Inside the equations two Boolean operators are used: $\oplus$ denotes the logical XOR (to avoid confusion with arithmetic addition) and the simple multiplication of two variables ab denotes the logical AND of a and b.

8.2.2 Ripple-Carry Adder

In [KRJ14] the authors presented a way to securely add two Boolean-masked values. Instead of three conventional steps (conversion, addition, reconversion), the addition can be implemented in just one step. Depending on the application, this can significantly increase the performance over the classical approach. The algorithm introduced in [KRJ14] is based on an Ripple Carry Adder (RCA). This adder has been rewritten into a sequence of Boolean operations that take the Boolean masks into consideration. The algorithm is word-oriented for efficiency in software but not in hardware.

Similarly, our design is based on the basic algorithm described in [KRJ14]. The underlying algorithm builds on the fact that one bit of sum s can be computed as

$$s_i = a_i \oplus b_i \oplus c_i\ . \tag{8.1}$$

Therefore, the addition is replaced by a simple XOR of the two operands a and b and the carry c. The only unknown part in such an equation is the carry bit which can be computed using a recursive formula

$$c_{i+1} = a_ib_i \oplus a_ic_i \oplus b_ic_i, \tag{8.2}$$

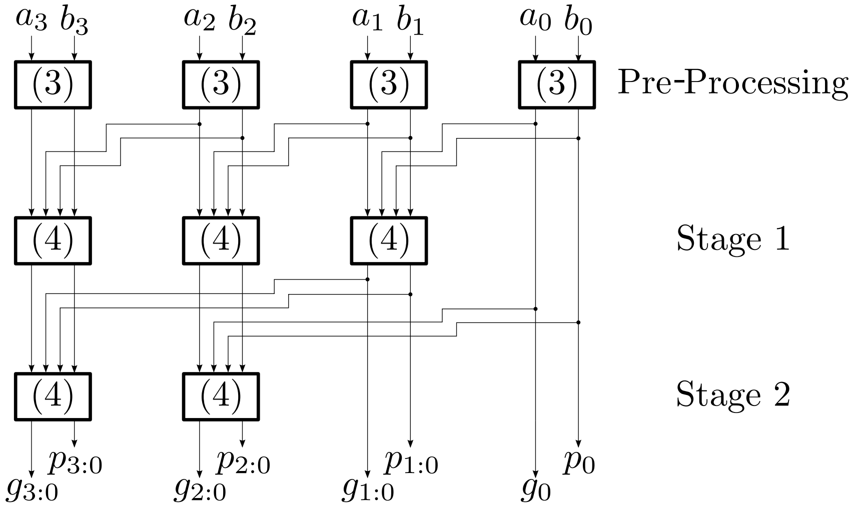


Figure 8.1: The structure of the carry generation for 4-bit operands using the KSA

where $c_0 = 0$. The costly part of the RCA is the recursive carry computation. Its function has to be evaluated iteratively which leads to a high circuit depth in case of a fully combinatorial design.

8.2.3 Kogge-Stone Adder

Another addition circuit with a lower depth is given by the Kogge-Stone Adder (KSA) [KS73] that splits the carry generation into generate (g) and propagate (p) functions. Instead of evaluating the carry function recursively, the KSA benefits from a tree-like structure and achieves a logarithmic complexity. For a hardware design, a KSA can significantly increase the overall performance.

The basic structure of KSA for $n = 4$-bit operands is shown in Figure 8.1. For operands a and b it computes the carry bits in three steps. During preprocessing the initial g_i and p_i values are generated as

$$g_i = a_i b_i \ , \qquad\qquad p_i = a_i \oplus b_i \ . \tag{8.3}$$

In the following stages a function is used to combine the g and p values of different bit positions. This function receives 4 bits as input and returns 2 output bits. For $i > j$ the output values are computed as

$$g_{i:j} = g_i \oplus g_j p_i \ , \qquad p_{i:j} = p_i p_j \ . \tag{8.4}$$

After $\log_2(n = 4) = 2$ stages, the computation is finished and all carry bits can be derived as

$$c_{i \in \{2 \ldots n\}} = g_{i-1:0} \ , \qquad c_1 = g_0 \ , \qquad c_0 = 0.$$

Finally, the sum s can be obtained according to Equation (8.1).

8.3 Implementation

We present two designs of a modulo 2^{32} adder that provides resistance against first- and second-order SCA. This is a quite common type of addition used in many cryptographic algorithms (e.g., Salsa20, HC-128, SHA-2), but our architectures can be also easily adapted to other bit lengths.

8.3.1 Ripple-Carry Adder (First-Order SCA-Resistant)

Based on the scheme presented in Section 8.2.2 we build a first-order SCA-resistant adder. To achieve this, Equations (8.1) and (8.2) should be transformed to meet the three required TI properties.

Given that Equation (8.2) is of degree 2, at least 3 shares (for input as well as for output) are necessary. It is supposed that each processed value, e.g., a_i, is split into 3 shares as (a_i^1, a_i^2, a_i^3). In case of Equation (8.1), due to its linearity the shares are easily combined via XOR as

$$s_i^1 = a_i^1 \oplus b_i^1 \oplus c_i^1 \ , \qquad s_i^2 = a_i^2 \oplus b_i^2 \oplus c_i^2 \ , \qquad s_i^3 = a_i^3 \oplus b_i^3 \oplus c_i^3 \ . \tag{8.5}$$

As mentioned before, Equation (8.2) is non-linear and has algebraic degree of 2. Following *direct sharing* approach represented in [BNN$^+$12], we can construct a correct and uniform shared implementation of such a function. The shares of the carry bit can be computed as

$$c_{i+1}^1 = a_i^2 b_i^2 \oplus a_i^2 b_i^3 \oplus a_i^3 b_i^2 \oplus a_i^2 c_i^2 \oplus a_i^2 c_i^3 \oplus a_i^3 c_i^2 \oplus b_i^2 c_i^2 \oplus b_i^2 c_i^3 \oplus b_i^3 c_i^2 \tag{8.6}$$

$$c_{i+1}^2 = a_i^3 b_i^3 \oplus a_i^3 b_i^1 \oplus a_i^1 b_i^3 \oplus a_i^3 c_i^3 \oplus a_i^3 c_i^1 \oplus a_i^1 c_i^3 \oplus b_i^3 c_i^3 \oplus b_i^3 c_i^1 \oplus b_i^1 c_i^3 \tag{8.7}$$

$$c_{i+1}^3 = a_i^1 b_i^1 \oplus a_i^1 b_i^2 \oplus a_i^2 b_i^1 \oplus a_i^1 c_i^1 \oplus a_i^1 c_i^2 \oplus a_i^2 c_i^1 \oplus b_i^1 c_i^1 \oplus b_i^1 c_i^2 \oplus b_i^2 c_i^1 \tag{8.8}$$

We note that Equations (8.1) and (8.2) can be seen as a function $f : (a_i, b_i, c_i) \mapsto (s_i, c_{i+1})$. At a first glance one may think of examining the uniformity of the (s_i, c_{i+1}) tuple[1]. However, such a tuple is never supplied to any function within the RCA algorithm. Note that s_i is an output bit and is not propagated while c_{i+1} is given to the next stage where it is combined with a_{i+1} and b_{i+1} which are independent of c_{i+1}. Hence, the uniformity of c_{i+1} suffices to fulfill the corresponding property.

During the implementation of such a design we encountered an issue that has never been reported before. The output of the shared carry computation function (Equation (8.6) to (8.8)) cannot be directly used as feedback signal since the output of a function from a previous cycle is used as input in the next clock cycle.

As a remedy we constructed a two-stage design as depicted in Figure 8.2. The three shares of the two operands a and b are stored in shift registers. The RCA algorithm and the deployment of shift registers supports an efficient scanning of operand bits. Two instances of the shared carry computation function are implemented whose outputs are stored in carry registers $\mathtt{c}_0$ and $\mathtt{c}_1$. The carry registers are enabled alternately while the other intermediate registers ($\mathtt{c}'_0$, $\mathtt{a}'_0$, $\mathtt{a}'_1$, $\mathtt{b}'_0$ and $\mathtt{b}'_1$) are enabled every second clock cycle synchronized with that of $\mathtt{c}_1$. The operand registers are also shifted two bits every other second clock cycle. Additional registers, i.e., $\mathtt{c}'_0$, $\mathtt{a}'_0$, $\mathtt{a}'_1$, $\mathtt{b}'_0$, and $\mathtt{b}'_1$, synchronize the computation of the sum bits, which needs to be performed one clock cycle after that of the carry bits. Note that we use the shift register of operand a to save the result of the addition.

Another issue is related to the first stage, i.e., when $i = 0$. In our designs we suppose that input carry $c_0 = 0$ so that (c_0^1, c_0^2, c_0^3) should be a shared representation of 0. Therefore, both

---

[1] If sharing of x and y are uniform, the tuple of sharing of (x, y) is not necessarily uniform if x and y are not independent.

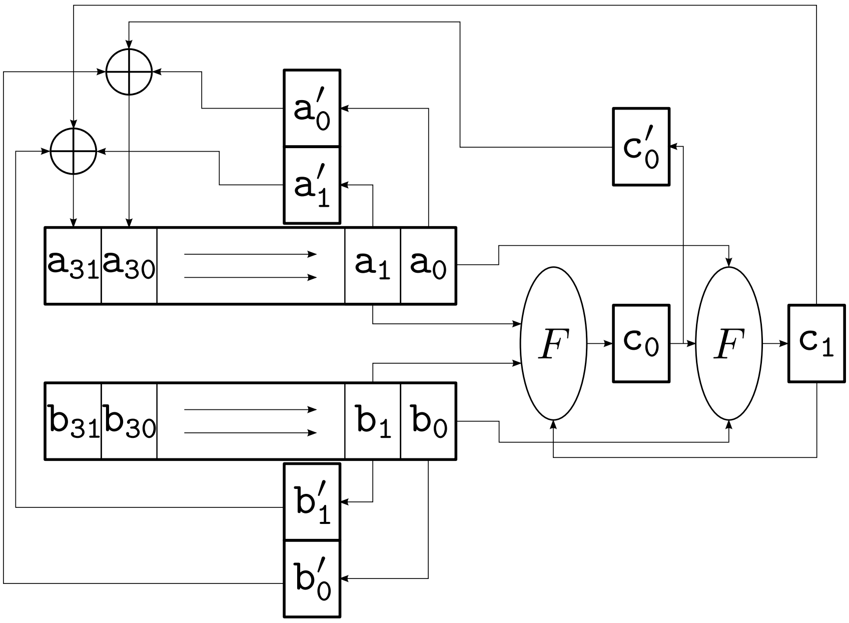


Figure 8.2: Structure of the first-order secure adder based on RCA.

carry registers have to be initialized with a random set representing 0. In other words, our design requires four fresh mask bits fm_1, ..., fm_4 only at the start of the addition to initialize $\mathtt{c}_0$ and $\mathtt{c}_1$ with $(fm_1, fm_2, fm_1 + fm_2)$ and $(fm_3, fm_4, fm_3 + fm_4)$ respectively. Note that all other stages of our design do not require fresh random bits leading to an efficient design with respect to the number of required fresh mask bits. For instance, our design is considerably more efficient than the solutions proposed in [KRJ14], [CGV14] and [CGTV15].

8.3.2 Ripple-Carry Adder (Second-Order SCA-Resistant)

The design described above can be simply transformed to support resistance against higher-order attacks. We now present a solution for the second-order resistant design. We increase the number of input shares s_{in} to 5 and all corresponding functions have to be chosen according to the principles of (univariate) higher-order TI [BGN⁺14b].

Equation (8.5) just needs to be adapted to the increased number of shares. The computation of the carry has to be split up into two steps. In the first step, $s_{out} = 10$ component functions generate 10 output shares. Following the same concept presented in [BGN⁺14b], these intermediate shares are then again reduced to 5 shares in the second step.

No major changes to the basic structure depicted in Figure 8.2 are necessary for implementation. Just the registers have to be adjusted to the increased number of shares and the F block is split by an additional register stage. All uniform equations for the F function, obtained by direct sharing, are described in detail in Appendix 13.1.

As a consequence, the amount of utilized resources increases and the number of clock cycles needed for the carry computation doubles. Just as before, the carry registers need fresh randomness during the initialization. Therefore, the number of required fresh random masks increases to 8 bits.

8.3.3 Kogge-Stone Adder (First-Order SCA-Resistant)

The design based on the RCA has low requirements for space and randomness. However, the number of clock cycles for one addition grows linearly with the bit length of the operands. For increased performance we therefore implemented a design that uses a KSA as foundation and which is still secure against first-order attacks.

Equations (8.3) and (8.4) need to be split into shares. Since all the corresponding formulas are of degree two, similar to that of the RCA, at least 3 shares are required to realize a functional TI.

The two outputs of the preprocessing step are both given to the next stages; thus, the uniformity of each tuple (g_i, p_i) must be taken into account. One part of Equation (8.3) needs to be implemented by the AND of the two operands for which no uniform TI with 3 shares exists [NRS11]. For this, fresh mask bits have to be used to make it uniform (see remasking in [MPL$^+$11, BNN$^+$12]). In our design, we adopted the solution from [BNN$^+$12] with only a single virtual share. One fresh random bit m_i is required for every invocation of the function in the preprocessing step:

$$g_i^1 = a_i^2 b_i^2 \oplus a_i^2 b_i^3 \oplus a_i^3 b_i^2 \oplus m_i \tag{8.9}$$

$$g_i^2 = a_i^3 b_i^3 \oplus a_i^1 b_i^3 \oplus a_i^3 b_i^1 \oplus a_i^1 m_i \oplus b_i^1 m_i \tag{8.10}$$

$$g_i^3 = a_i^1 b_i^1 \oplus a_i^1 b_i^2 \oplus a_i^2 b_i^1 \oplus a_i^1 m_i \oplus b_i^1 m_i \oplus m_i\ . \tag{8.11}$$

Further, preprocessing involves another linear XOR-function. We implement this part similar to Equation (8.5). Both functions and their joint output (g_i, p_i) fulfill the three TI properties.

The preprocessing step is followed by stages in which the g and p values are updated according to Equation (8.4). These two functions can be considered as a 4-bit to 2-bit mapping. Similar to the preprocessing step, the tuple of the 2-bit output has to be considered for the uniformity check. For the computation of the g part of Equation (8.4) (as an AND/XOR operation), we followed the *direct sharing* approach [BNN$^+$12] and achieved:

$$g_{i:j}^1 = g_i^2 \oplus g_j^2 p_i^2 \oplus g_j^2 p_i^3 \oplus g_j^3 p_i^2 \tag{8.12}$$

$$g_{i:j}^2 = g_i^3 \oplus g_j^3 p_i^3 \oplus g_j^1 p_i^3 \oplus g_j^3 p_i^1 \tag{8.13}$$

$$g_{i:j}^3 = g_i^1 \oplus g_j^1 p_i^1 \oplus g_j^1 p_i^2 \oplus g_j^2 p_i^1\ . \tag{8.14}$$

The other part (computation of p) of Equation (8.4) can be implemented similar to Equation (8.9). To reduce the amount of required fresh random bits, we replaced m_i with g_j^1. This bit is not used in this equation and can take the role of a mask. Although our construction does not closely follow the assumptions in [BNN$^+$12] considering the construction of virtual shares, we can demonstrate that this has no impact on security. Our simulation results show not only the uniformity of shared $p_{i:j}$ but also the uniformity of the shared tuple $(p_{i:j}, g_{i:j})$. We need to emphasize that due to the specific architecture of the KSA algorithm, g_j^1 is only used once as a mask to introduce uniformity into the computation of a p. In other words, the mask bit g_j^1 is never reused again what could potentially violate the uniformity in later stages.

Our design is optimized for maximum throughput by using a fully pipelined architecture. Figure 8.3 depicts the basic structure of our design. Since only the preprocessing step requires fresh random bits and – as stated above – all other stages are computed without additional mask, the total number of required fresh mask bits is $n = 32$. Compared to the other solutions like [KRJ14], [CGV14] and [CGTV15] this is still reasonable.

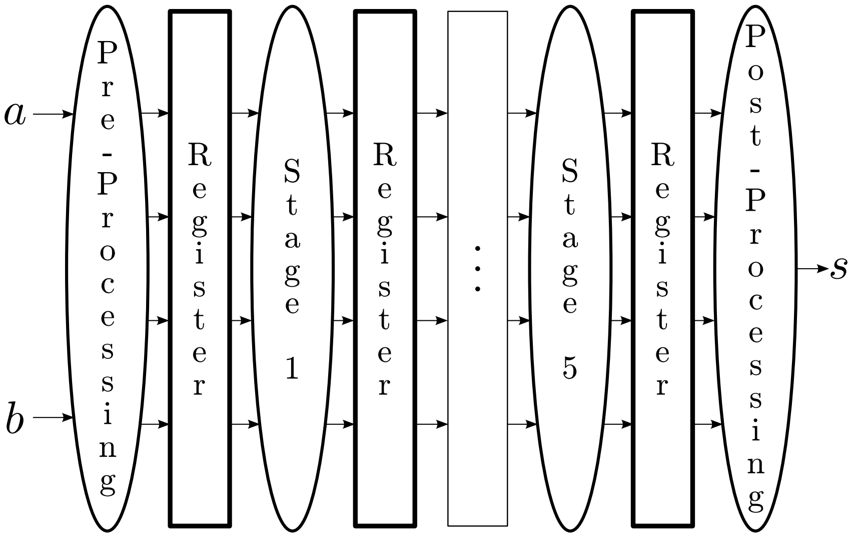


Figure 8.3: Block diagram of the first-order secure adder based on KSA.

8.3.4 Kogge-Stone Adder (Second-Order SCA-Resistant)

Similar as for the RCA, the design based on the KSA is also easily extensible to higher orders. We outline the exemplary procedure for second-order security. In this case, the number of input shares is again set to 5. The four aforementioned equations are adjusted to meet the requirements of the second-order TI.

The XOR part of Equation (8.3) is implemented as before but adapted to the higher number of shares. The AND part to compute g_i of Equation (8.3) is split into two steps. As before, the first step results in 10 output shares and the second step merges the last 6 shares into a total number of 5 shares again. Furthermore, we have to use fresh masks to assure the uniformity. In this case, four fresh random bits are necessary. The two functions of the following stages are also split into two steps. For the computation of $g_{i:j}$ of Equation (8.4) we use the second-order TI representation of the AND/XOR function given in [BGN$^+$14b]. For the $p_{i:j}$ part (the AND operation) we use the same construction of g_i of the preprocessing step. Instead of four fresh mask bits, we used 4 shares of g_j as fresh masks to reduce the required randomness. Details of the underlying uniform equations can be found in Appendix 13.2.

The basic structure as shown in Figure 8.3 is also the template for the architecture of most other parts. It has mainly to be adapted to 5 shares and the functions need to be split into two steps with a register in between. Hence, the number of clock cycles for one addition is doubled. In terms of randomness the demand of our implementation quadruples, because each invocation of the AND operation in the preprocessing step requires 4 random bits.

8.3.5 Comparison

We now compare our designs in terms of size and performance that are implemented on a Spartan-6 FPGA with other solutions. All our findings are summarized in Table 8.1. In terms of size, the RCA-based variant is clearly superior to other solutions due to the iterative structure.

Table 8.1: Results and comparison of our hardware architectures.

	LUTs	*FFs*	*Latency* (CLK)	*Frequency* (MHz)	*Throughput* (Mbit/s)	*Randomness* (bit)
RCA *(1st order)*	227	223	32	101	101	4
RCA *(2nd order)*	388	387	65	107	52	$8{=}4 \cdot d$
KSA *(1st order)*	937	1330	6	62	330	32
KSA *(2nd order)*	4223	5509	12	63	168	$28{=}2 \cdot d \cdot n$
[KRJ14] *(1st order)*	-	-	-	-	-	n
[CGV14] *(dth order)*	-	-	-	-	-	$(2 \cdot d^2 + d) \cdot n$
[CGTV15] *(1st order)*	-	-	-	-	-	$3 \cdot n$

On the contrary, the designs based on the KSA provide low latency and high-performance applications.

Due to the different implementation platforms, we cannot fairly compare our hardware designs with software-based solutions [KRJ14], [CGV14] and [CGTV15]. Therefore, Table 8.1 restricts the comparison to the number of required fresh random bits.

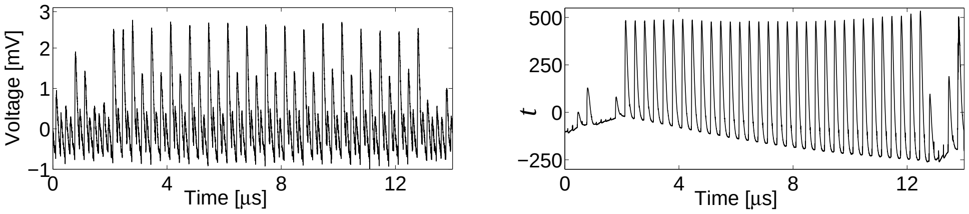
We can conclude that the RCA-based design is most efficient regarding the number of required random bits. The requirement of $4 \cdot d$ random bits outperforms all other proposals and is also independent of the operands bit length n. The approach based on KSA requires a higher number of random bits which also depends on n. Nevertheless, the first-order secure design uses the same amount of fresh masks as the solution of [KRJ14] and less than[CGTV15]. For higher orders it even outperforms the design of [CGV14]. It is noteworthy that the number of fresh masks for d-order KSA with $d \geq 2$ can be decreased even further. For $d = 1$ we can use the trick presented in [BNN$^+$12] that requires only one fresh mask bit for an AND operation. Such a construction – with one virtual variable – might be also found for higher-order TI of the AND operation thereby reducing the number of required fresh mask bits.

8.4 Analysis

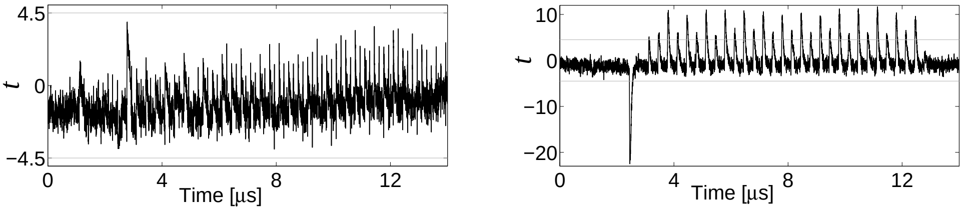
For the practical SCA evaluations we employed a SAKURA-G platform [Sak] populated with a Spartan-6 FPGA as target. All SCA traces have been collected by a Digital Storage Oscilloscope (DSO) while measuring the voltage drop over a $1\,\Omega$ resistor in Vdd path. In order to obtain clean signals and reduce the electrical noise, we used the embedded amplifier of the SAKURA-G and restricted the bandwidth of the oscilloscope to 20 MHz. As evaluation metric, we applied a *non-specific* statistical t-test [GJJR11].

8.4.1 Ripple-Carry Adder

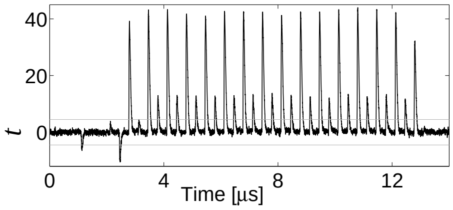
Now we analyze the security of our first-order SCA-resistant RCA design. A sample trace of such a design is shown in Figure 8.4(a). In order to have a reference for the existing leakage in our platform as well as an evidence for the suitability of the applied evaluation scheme, we first turned the PRNG off that provides the randomness for initial sharing and fresh masks. Hence, all outputs of the PRNG are set to zero and the underlying design receives unshared inputs as



(a) sample trace (b) PRNG inactive



(c) PRNG active (1st order) (d) PRNG active (2nd order)



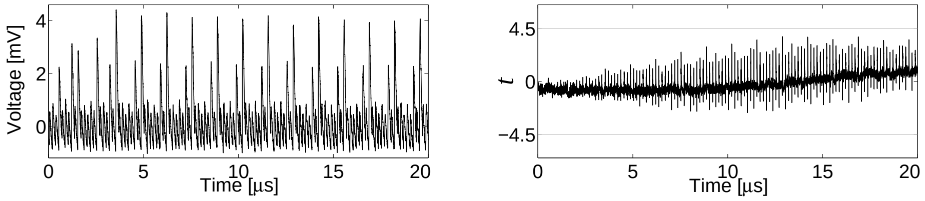
(e) PRNG active (3rd order)

Figure 8.4: RCA 1st order, t-test results using 100 000 000 traces.

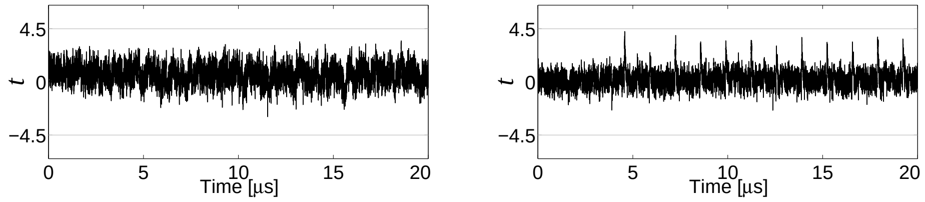
$(a, 0, 0)$ and $(b, 0, 0)$. With such a setting we collected 100 000 traces corresponding to a mixture of fixed and random inputs. Therefore, we expect the t-test to report clearly exploitable first-order leakages, which is confirmed by the corresponding result shown in Figure 8.4(b). It can be seen that the t value exceeds 400 during the cryptographic operation exceeding the defined threshold by far.

As the next step we activated the PRNG so that the adder circuit receives randomly shared inputs and fresh random masks. Hence, the design is expected to provide first-order security. In order to examine this we collected 100 000 000 traces and performed the t-test up to third order. The corresponding results shown in Figure 8.4 indicate the resistance of the design to first-order attacks and – as expected – its vulnerability to second- and third-order attacks.

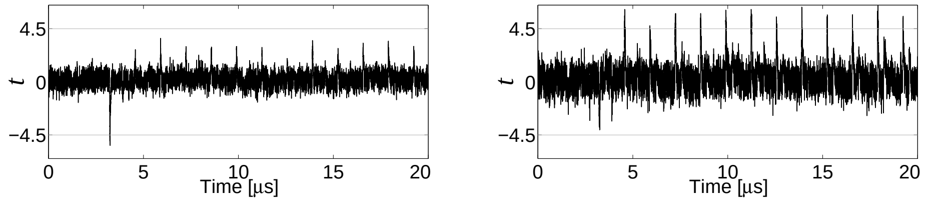
We continued our evaluation with the second-order-SCA-resistant RCA design with an active PRNG. Due to the high amount of randomness, i.e., fourth-order Boolean masking (five shares), exploiting a leakage from such a design needs a large number of traces. Therefore, following the above-explained procedure we collected 300 000 000 traces and ran the t-test evaluations. The results shown in Figure 8.5 confirm the resistance of our design to first- and second-order attacks. Similar to the results of [BGN^{+}14b], the third-order leakage still cannot be detected, but we observe fourth- and fifth-order leakages as the design with five shares is expected to be vulnerable to a fifth-order attack.



(a) sample trace (b) PRNG active (1st order)

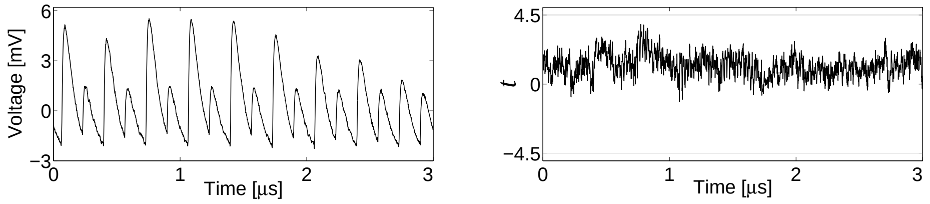


(c) PRNG active (2nd order) (d) PRNG active (3rd order)

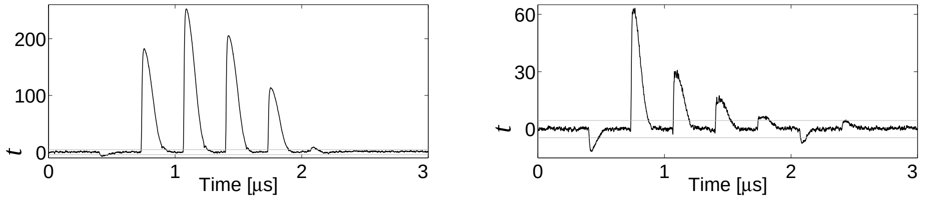


(e) PRNG active (4th order) (f) PRNG active (5th order)

Figure 8.5: RCA 2nd order, t-test results using 300 000 000 traces.

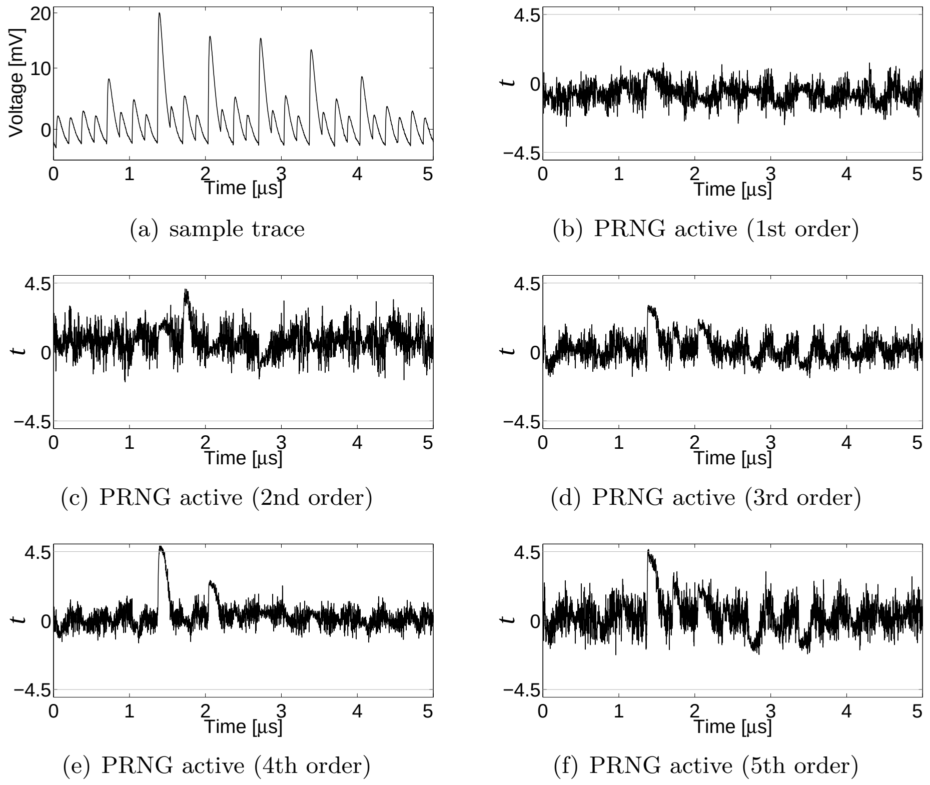


(a) sample trace (b) PRNG active (1st order)



(c) PRNG active (2nd order) (d) PRNG active (3rd order)

Figure 8.6: KSA 1st order, t-test results using 100 000 000 traces.



(a) sample trace (b) PRNG active (1st order)

(c) PRNG active (2nd order) (d) PRNG active (3rd order)

(e) PRNG active (4th order) (f) PRNG active (5th order)

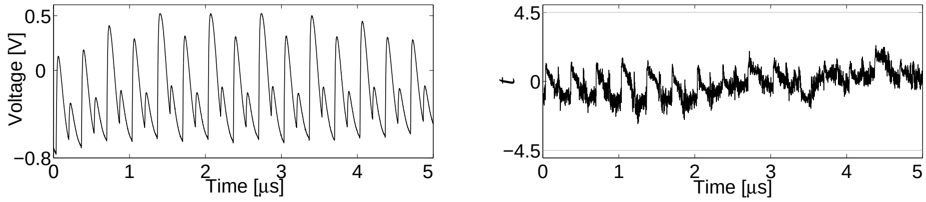
Figure 8.7: KSA 2nd order, t-test results using 300 000 000 traces.

8.4.2 Kogge-Stone Adder

Both analyses on the first- and second-order RCA were repeated on the first- and second-order SCA-resistant KSA designs. We even collected the same number of traces, i.e., 100 000 000 traces to evaluate the first-order KSA and 300 000 000 traces for the second-order KSA. The results which confirm the resistance of our constructions are shown in Figure 8.6 and Figure 8.7, respectively.

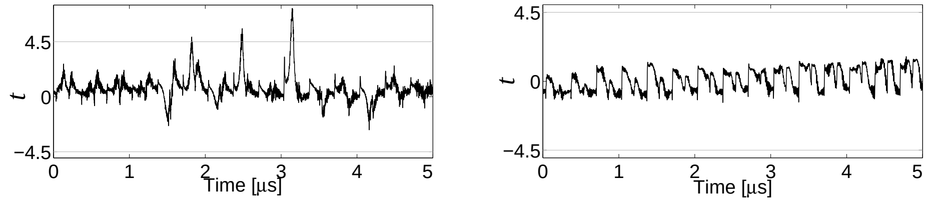
8.4.3 Higher-Order Security

Shortly after the initial publication of high-order threshold implementations [BGN$^+$14b], Reparaz published a note [Rep15] on the security of higher-order threshold implementations which was extended in [RBN$^+$15]. It states that when different intermediates values, i.e., shares, from different clock cycles are combined, a second-order TI might be vulnerable to the corresponding second-order attack. Although confirming this statement in general, we like to emphasize that this is not addressed in [BGN$^+$14b]. In the model of univariate higher-order attacks, all lemmas and proofs as given in [BGN$^+$14b] remain valid. Furthermore, this is backed by our practical investigations as shown above. Still, we need to highlight that the second-order TI designs we presented in this work are designed to resist against univariate second-order attacks.



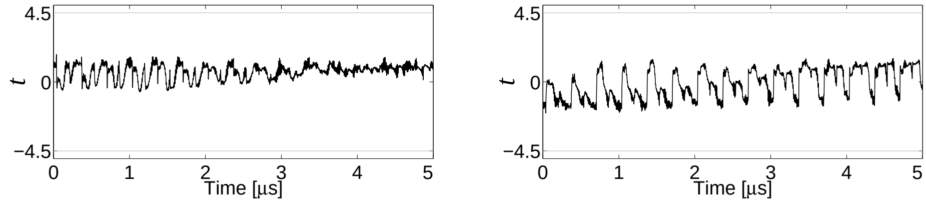
(a) sample trace

(b) PRNG active (1st order)



(c) PRNG active (2nd order)

(d) PRNG active (3rd order)



(e) PRNG active (4th order)

(f) PRNG active (5th order)

Figure 8.8: (modified measurement setup) KSA 2nd order, t-test results using 300 000 000 traces.

In this context, it has been previously shown in [MM13] that multivariate leakages can be easily summed up and be represented in a univariate form. The suggested approaches for such a combination include (a) running the target device at a relatively high frequency, e.g., 24 MHz - 48 MHz, and (b) making use of a Direct Current (DC) blocker and/or certain amplifiers in the measurement setup. Both techniques cause overlapping the power peaks of adjacent clock cycles, and hence the leakage associated to consecutive clock cycles are somehow added together. In [MM13] it has been shown that employing any of the aforementioned techniques causes an implementation of a univariate second-order resistant design to be vulnerable to a univariate second-order attack.

In order to examine the effect of such an issue on our second-order TI designs, we considered the second aforementioned technique. In other words, we employed a DC blocker (BLK-89-S+ from Mini-Circuits) and two serially connected AC amplifiers (ZFL-1000LN+ from Mini-Circuits) in the measurement setup. By means of this setup we repeated the same measurements and evaluations of our developed second-order Kogge-Stone Adder using the same number of 300 000 000 traces. We kept the measurement settings, e.g., sampling rate, bandwidth, and the target frequency of operation, the same as the last experiments. The results shown in Figure 8.8 indeed practically confirm the note given in [Rep15]. The second-order TI design demonstrates second-order leakages when the power peak of consecutive clock cycles are combined (by the measurement setup). Interestingly, by such a measurement setup the 4th-order and 5th-order analyses (in contrary to the previous experiment of Figure 8.7) do not show a detectable leakage. We believe that it is due to the noise introduced by the measurement setup, i.e., overlapping the adjacent power peaks, which can certainly affect the feasibility of higher-order attacks.

8.5 Conclusion and Future Work

In this paper, we presented two ways of performing addition on Boolean-masked values that are secure against SCA attacks on a hardware platform. Compared to the KSA-based approach, the RCA-based solution is slower but requires less space and the least amount of random bits. In terms of performance, the design based on the KSA provides a suitable choice due to its pipelined architecture. In comparison to other already published algorithms, our approaches are able to match and even reduce the randomness requirements especially for higher orders. The resistance of both approaches has been verified by practical evaluations showing the security of our constructions. Our proposed designs enable an efficient and secure implementation of ARX-based designs in hardware which have not been fully investigated yet.

Thus, our designs can be seen as the foundation for future work involving the masked implementation of ARX-based algorithms, e.g., ChaCha [Ber08a]. Furthermore, to achieve multivariate higher-order security our high-order TI design could be extended using the principle from [RBN+15]. Another interesting direction for future research is related to the secure conversion of arithmetic-masked to Boolean-masked values. Similar to [CGTV15], which includes both conversion and blinded addition algorithms, our main design does not need to be fundamentally changed to provide the conversion. We only need to include additional random masks and adjust the input parameters.

Chapter 9

Strong 8-bit Sboxes with Efficient Masking in Hardware

Block ciphers are arguably the most important cryptographic primitive in practice. While their security against mathematical attacks is rather well understood, physical threats such as SCA still pose a major challenge for their security. An effective countermeasure to thwart SCA is using a cipher representation that applies the threshold implementation (TI) concept. However, there are hardly any results available on how this concept can be adopted for block ciphers with large (i.e., 8-bit) Sboxes. In this chapter we provide a systematic analysis based on [BGG$^+$16a] and search for 8-bit Sbox constructions that can intrinsically feature the TI concept, while still providing high resistance against cryptanalysis. Our study includes investigations on Sboxes constructed from smaller ones using Feistel, Substitution-Permutation Network (SPN), or MISTY network structures. As a result, we present a set of new Sboxes that not only provide strong cryptographic criteria, but are also optimized for TI. We believe that our results will found an inspiring basis for further research on high-security block ciphers that intrinsically feature protection against physical attacks.

Contents of this Chapter

9.1 Introduction **105**
9.2 Background **107**
9.3 Threshold Implementation of Known 8-Bit Sboxes **111**
9.4 Finding TI-Compliant 8-Bit Sboxes **116**
9.5 Results **121**
9.6 Conclusion and Future Work **124**

9.1 Introduction

Block ciphers are among the most important cryptographic primitives. Although they usually follow ad hoc design principles, their security with respect to known attacks is generally well-understood. However, this is not the case for the security of their implementations. The security of an implementation is often challenged by physical threats such as side-channel analysis or fault-injection attacks. In many cases, those attacks render the mathematical security meaningless. Hence, it is essential that a cipher implementation incorporates appropriate countermeasures against physical attacks. Usually, those countermeasures are developed retroactively for a given, fully specified block cipher. A more promising approach is including the possibility of adding efficient countermeasures into the design from the very start.

For software implementations, this has been done. Indeed, a few ciphers have been proposed that aim to address the issue of protection against physical attacks by facilitating a masked Sbox by design. The first example is certainly NOEKEON [DPAR00], other examples include Zorro [GGNS13], Picarro [PRC12] and the LS-design family of block ciphers [GLSV14].

For hardware implementations, the situation is significantly different. Here, simple masking is less effective due to several side-effects, most notably glitches (see [MPO05]). Unfortunately, it is not trivial to apply the TI concept to a given block cipher. The success of this process strongly depends on the complexity of the cipher's round function and its internal components. While the linear aspects of any cipher are typically easy to convert to TI, this is not generally true for the non-linear Sbox. For 4-bit Sboxes, it is possible to identify a corresponding TI representation by exhaustive search [BNN$^+$12]. However, for larger Sboxes, in particular 8-bit Sboxes, the situation is very different. In this case, the search space is far too large to allow an exhaustive search. In fact, 8-bit Sboxes are far from being fully understood, from both a cryptographic and an implementation perspective.

With respect to cryptographic strength against differential and linear attacks, the AES Sbox (and its variants) can be seen as holding the current world record. We do not know of any Sbox with better properties, but those might well exist. Unfortunately, despite considerable effort, no TI representation is known for the AES Sbox that does not require any additional external randomness [BGN$^+$14a, BGN$^+$15, MPL$^+$11].

9.1.1 Contribution

In this chapter we approach this problem of identifying cryptographically strong 8-bit Sboxes that provide a straightforward TI representation. More precisely, our goal is to give examples of Sboxes that come close to the cryptanalytic resistance of the AES Sbox and the straight application of the TI concept to these Sboxes should still lead to minimal resource and area costs. This enables an efficient and low-cost implementation in hardware and in a bit-sliced fashion for software implementations.

In our work we systematically investigate 8-bit Sboxes that are constructed based on what can be seen as a mini-cipher. Concretely, we construct Sboxes based on either a Feistel-network (operating with two 4-bit branches and a 4-bit Sbox as the round function), a substitution permutation network or the MISTY network. This general approach has already been used and studied extensively. Examples of Sboxes constructed like this are used for example in the ciphers Crypton [Lim98, Lim99], ICEBERG [SPR$^+$04], Fantomas [GLSV14], Robin [GLSV14] and Khazad [BR00]. A more theoretical study was most recently presented in [CDL15].

Our idea extends the previous work by combining those constructions aiming at achieving strong cryptographic criteria with small Sboxes that are easy to share and intrinsically support the TI concept. As a result of our investigation, we present a set of different 8-bit Sboxes. These Sboxes are either (a) superior to the known constructions from a cryptographic perspective but can still be implemented with moderate resource requirements or (b) outperform all known constructions in terms of efficiency in the application of the TI concept to the Sbox, while still maintaining a comparable level of cryptographic strength with respect to other known Sboxes. All our findings are detailed in Table 9.2.

9.2 Background

In this section we recall the basic (cryptographic) properties of Sboxes and the basic principles of their implementation in hardware.

9.2.1 Cryptanalytic Properties of Sboxes

In this subsection we recall the tools used for evaluating the strength of Sboxes with respect to linear, differential and algebraic properties. For this purpose, we consider an n-bit Sbox S as a vector of Boolean functions: $S = (f_0, \dots, f_{n-1})$, $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$. We denote the cardinality of a set A by $\#A$ and the dot product between two elements $a, b \in \mathbb{F}_2^n$ by: $\langle a, b\rangle = \sum_{i=0}^{n-1} a_i b_i$.

Non-Linearity

To be secure against linear cryptanalysis [Mat93] a cipher must not be well-approximated by linear or affine functions. As the Sbox is generally the only non-linear component in an SP-network, it has to be carefully chosen to ensure a design is secure against linear attacks. For a given Sbox, the main criterion here is the Hamming distance of any component function, i.e. a linear combination of the f_i, to the set of all affine functions. The greater this distance, the stronger the Sbox with respect to linear cryptanalysis. The Walsh transform $W_S(a, b)$, defined as

$$W_S(a,b) := \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle a,x\rangle + \langle b,S(x)\rangle},$$

can be used to evaluate the correlation of a linear approximation $(a, b) \neq (0, 0)$. More precisely,

$$\mathrm{P}(\langle b, S(x)\rangle = \langle a, x\rangle) = \frac{1}{2} + \frac{W_S(a,b)}{2^{n+1}}.$$

The larger the absolute value of $W_S(a, b)$, the better the approximation by the linear function $\langle a, x\rangle$ (or the affine function $\langle a, x\rangle + 1$, in case $W_S(a, b) < 0$).

This motivates the following well-known definition.

Definition 9.2.1. Given a vectorial Boolean function S, its *linearity* is defined as

$$\mathrm{Lin}(S) = \max_{a,b \neq 0} |W_S(a,b)|.$$

The smaller $\mathrm{Lin}(S)$, the stronger the Sbox is against linear cryptanalysis.

It is known that for any function S from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ it holds that $\mathrm{Lin}(S) \geq 2^{\frac{n+1}{2}}$ [CV94]. Functions that reach this bound are called Almost Bent (AB) functions. However, in the case $n > 4$ and n even, we do not know the minimal value of the linearity that can be reached. In particular, for $n = 8$ the best known non-linearity is achieved by the AES Sbox with $\mathrm{Lin}(S) = 32$.

Differential Uniformity

A cipher must also be resistant against differential cryptanalysis [BS90]. To evaluate the differential property of an Sbox, we consider the set of all non-zero differentials and their probabilities (up to a factor 2^{-n}). That is, given $a, b \in \mathbb{F}_2^n$ we consider

$$\delta_S(a, b) := \#\{x \in \mathbb{F}_2^n \| \; S(x + a) = S(x) + b\},$$

which corresponds to 2^n times the probability of an input difference a propagating to an output difference b through the function S. This motivates the following well-known definition.

Definition 9.2.2. Given a vectorial Boolean function S, its *differential uniformity* is defined as

$$\mathrm{Diff}(S) = \max_{a \neq 0, b} |\delta_S(a, b)|.$$

The smaller $\mathrm{Diff}(S)$, the stronger the Sbox regarding differential cryptanalysis.

It is known that for Sboxes S that have the same number of input and output bits it holds that $\mathrm{Diff}(S) \geq 2$. Functions that reach that bound are called Almost Perfect Nonlinear (APN). While APN functions are known for any number n of input bits, APN *permutations* are known only in the case of n odd and $n = 6$.

In particular, for $n = 8$ the best known case is $\mathrm{Diff}(S) = 4$, e.g., AES Sbox.

Algebraic Degree

The algebraic degree is generally considered as a good indicator of security against structural attacks, such as integral, higher-order differential or, most recently, attacks based on the division property.

Recall that any Boolean function f can be uniquely represented using its Algebraic Normal Form (ANF):

$$f(x) = \sum_{u \in \mathbb{F}_2^n} a_u x^u,$$

where $x^u = \prod_{i=0}^{n-1} x_i^{u_i}$, with the convention $0^0 = 1$. Now, the algebraic degree can be defined as follows.

Definition 9.2.3. The algebraic degree of f is defined as:

$$\deg(f) = \max_{u \in \mathbb{F}_2^n} \left\{ \sum_i u_i, a_u \neq 0 \right\}.$$

This definition can be extended to vectorial Boolean functions (Sboxes) as follows

$$\deg(S) = \max_{0 \leq i \leq n} \deg(f_i).$$

For a permutation on $\mathbb{F}_2^n$ the maximum degree is $n - 1$. Lots of permutations over $\mathbb{F}_2^n$ achieve this maximal degree. Again the AES Sbox is optimal in this respect, i.e., the AES Sbox has the maximal degree of 7 for 8-bit permutations.

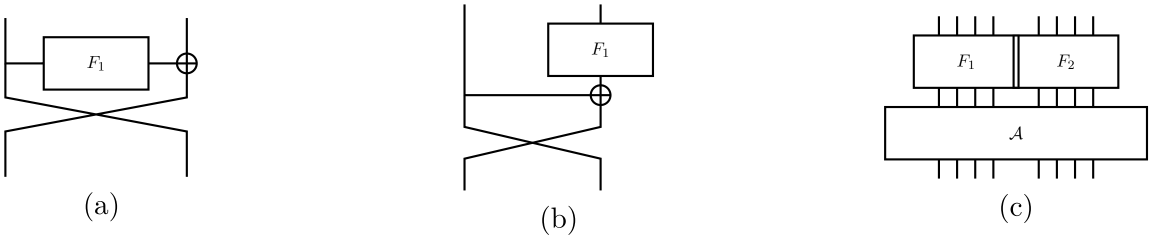


Figure 9.1: (a): Feistel (b) MISTY (c) SPN

Affine Equivalence

An important tool in our search for good Sboxes is the notion of affine equivalence. We say that two functions f and g are affine equivalent if there exists two affine permutations A_1 and A_2 such that $f = A_1 \circ g \circ A_2$. The importance of this definition is given by the well-known fact that both the linearity and the differential uniformity are invariant under affine equivalence. That is, two functions that are affine equivalent have the same linear and differential criteria.

9.2.2 Construction of 8-Bit Sboxes

Apart from the AES Sbox, which is basically the inversion in the finite field $\mathbb{F}_{2^8}$, hardly any primary construction for useful, cryptographically strong, 8-bit Sboxes is known.

However, several secondary constructions have been applied successfully. Here, the idea is to build larger Sboxes from smaller Sboxes . For block ciphers this principle was first introduced in MISTY [Mat97].

Later, this approach was modified and extended. In particular, it was used by several lightweight ciphers to construct Sboxes with different optimization criteria, e.g., smaller memory requirements, more efficient implementation, involution, and easier software-level masking.

There are basically three known constructions, all of which can be seen as mini-block ciphers: Feistel-networks, the MISTY construction and SP-networks. Figure 9.1 shows how these constructions build larger Sboxes from smaller Sboxes. Note that the MISTY construction is a special case of the SPN. Indeed, the MISTY construction is equivalent to SPN when $F_1 = Id$ and the matrix $\mathcal{A} = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$.

For a small number of rounds, we can systematically analyze the cryptographic properties of those constructions (see [CDL15] for the most recent results). However, for a larger number of rounds, a theoretical understanding becomes increasingly more difficult in most cases.

Table 9.2 shows the different characteristics of 8-bit Sboxes known in the literature that are built from smaller Sboxes. We excluded the PICARO Sbox [PRC12] from the list, since it is not a bijection. Furthermore, Zorro is also excluded since the exact specifications of its structure are not publicly known. We often refer to this table as it summarizes all our findings and achievements.

9.2.3 TI of 4-bit Permutations.

In [BNN$^+$15] the authors analyze 4-bit permutations and identify 302 equivalence classes. In the following, we use the same notation as in [BNN$^+$15] to refer to these classes. Out of these

Table 9.1: Performance figures of 4 × 4 quadratic bijections with respect to their TI cost.

	Table	Area [GE]	# of stages
$\mathcal{Q}^4_4$	`0123456789ABDCFE`	27	1
$\mathcal{Q}^4_{12}$	`0123456789CDEFAB`	63	1
$\mathcal{Q}^4_{293}$	`0123457689CDEFBA`	84	1
$\mathcal{Q}^4_{294}$	`0123456789BAEFDC`	51	1
$\mathcal{Q}^4_{299}$	`012345678ACEB9FD`	114	1
$\mathcal{Q}^4_{300}$	`0123458967CDEFAB`	151	2 ($\mathcal{Q}_{12} \circ \mathcal{Q}_4$)

302, six classes are quadratic. These six quadratic functions, whose uniform TI can be achieved by *direct sharing* or with simple *correction terms* (see [BNN$^+$15]) are listed in Table 9.1. We included their minimum area requirements as the basis of our investigations in the next sections. In contrast to the others, $\mathcal{Q}_{300}$ also needs to be decomposed for uniform sharing.

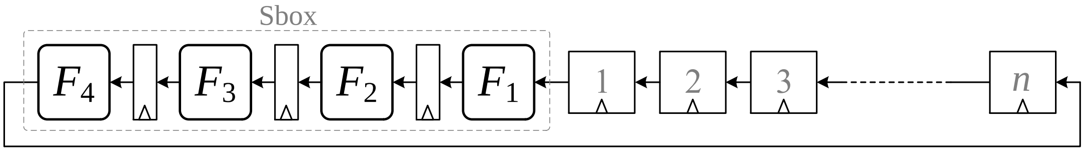
9.2.4 Design Architectures

Due to the high area overhead of threshold implementations (particularly the size of the shared Sbox), serialized architectures are favored, e.g. in [BGN$^+$15, MPL$^+$11, PMK$^+$11, STE15]. Our main target in this work is a serialized architecture in which one instance of the Sbox is implemented. Furthermore, we focus on byte-wise serial designs due to our underlying 8-bit Sbox target. In such a scenario, the state register forms a shift register, that at each clock cycle shifts the state bytes through the Sbox and makes use of the last Sbox output as feedback. Figure 9.2 depicts three different architectures which we can consider. Note that extra logic is not shown in this figure, e.g. the multiplexers to enable other operations like ShiftRows.

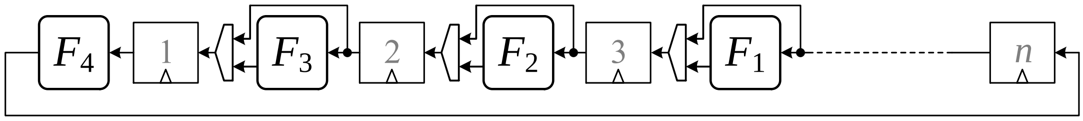
A shared Sbox with 3 shares should contain registers, e.g., PRESENT [PMK$^+$11] and AES [BGN$^+$15, MPL$^+$11]. As an example, if the shared Sbox contains 4 stages (see Figure 9.2(a)) and forms a pipeline, all the Sbox computations can be done in $n + 3$ clock cycles, with n as the number of state bytes. We refer to this architecture as *raw* in later sections. Note that realizing a pipeline is desirable. Otherwise, the Sbox computations would take $3n + 1$ clock cycles.

As an alternative, we can use the state registers as intermediate registers of the shared Sbox. Figure 9.2(b) shows the corresponding architecture, where more multiplexers should be integrated to enable the correct operation (as an example in Skinny [BJK$^+$16]). In this case, all n shared Sboxes can be computed in n clock cycles. It is noteworthy that such an optimization is not always fully possible if intermediate registers of the shared Sbox are larger than the state registers (e.g., in case of AES [BGN$^+$15, MPL$^+$11]).

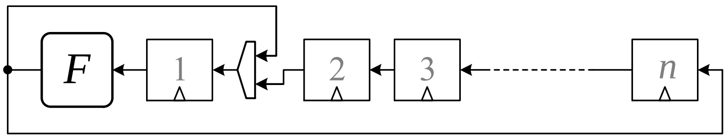
If the Sbox has been constructed by k times iterating a function F, it is possible to significantly reduce the area cost. Figure 9.2(c) shows an example. Therefore, similar to a raw architecture without pipeline, $(k-1)\,n+1$ clock cycles are required for n Sboxes. This is not efficient in terms of latency, but is favorable for low-throughput applications, where very low area is available and in particular when SCA protection is desired. We refer to this architecture as *iterative*.



(a) raw



(b) interleaved



(c) iterative

Figure 9.2: Different serialized design architectures.

9.3 Threshold Implementation of Known 8-Bit Sboxes

Among 8-bit Sboxes, the AES TI Sbox has been widely investigated while nothing about the TI of other Sboxes can be found in public literature. The first construction of the AES TI Sbox was reported in [MPL$^+$11]. The authors made use of the tower-field approach of Canright [Can05] and represented the full circuit by quadratic operations. By applying second-order Boolean masking, i.e., three shares as minimum following the TI concept, all operations are independently realized by TI. On the other hand, the interconnection between (and concatenation of) uniform TI functions may violate the uniformity. Therefore, the authors integrated several fresh random masks – known as remasking or applying virtual shares [BNN$^+$15] – to maintain the uniformity, in total 48 bits for each full Sbox. Since the AES TI Sbox has been considered for a serialized architecture, the authors formed a 4-stage pipeline design, which also increased the area by 138 registers.

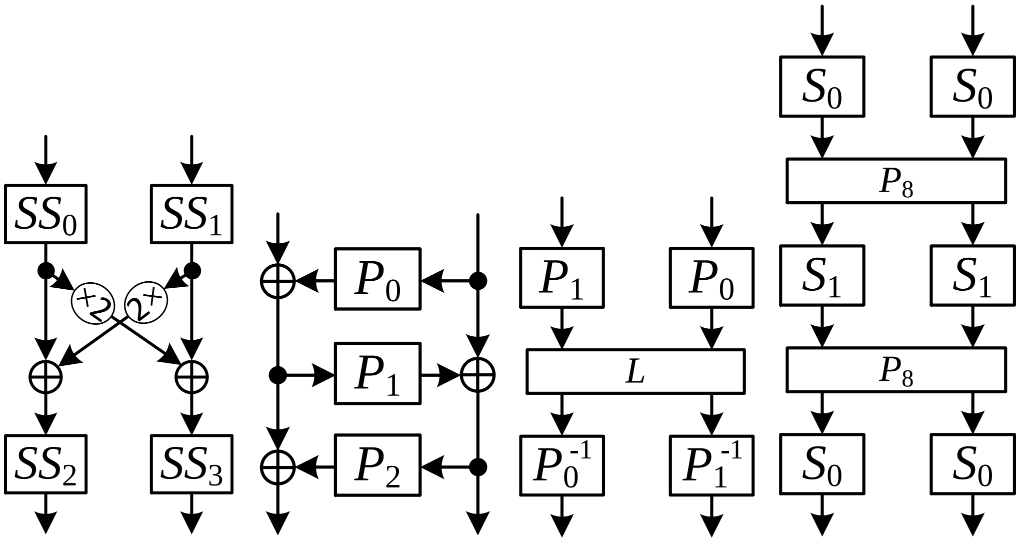
Later in [BGN$^+$15] three more efficient variants of the AES TI Sbox were introduced. The authors applied several tricks, e.g., increasing the number of shares to 4 and 5, and reduce them back to 3 in order to relax the fresh randomness requirements. Details of all different designs are listed in Table 9.2. In short, the most efficient design (called *nimble*) forms a 3-stage pipeline, where 92 extra registers and 32 fresh random bits are required.

9.3.1 CLEFIA

CLEFIA makes use of two 8-bit Sboxes S_0 and S_1 as depicted in Figure 9.3(a). The first one is formed by utilizing four different 4-bit bijections and multiplication by 2 in GF(2^4) defined by polynomial $X^4 + X + 1$. The entire SS_0 : `E6CA872FB14059D3`[1], SS_1 : `640D2BA39CEF8751`, SS_2 :

---

[1] In the following we denote functions by a hexadecimal-string in which the first letter denotes the first element of the look-up table implementing the function.



(a) CLEFIA (b) Crypton V0.5 (c) Crypton V1 (d) ICEBERG

Figure 9.3: Structure of the Sboxes of CLEFIA, Crypton V0.5, Crypton V1, and ICEBERG.

`B85EA64CF72310D9`, and SS_3 : `A26D345E0789BFC1` are cubic and – based on the classification given in [BNN+15] – belong to classes $\mathcal{C}_{210}$, $\mathcal{C}_{163}$, $\mathcal{C}_{160}$, and $\mathcal{C}_{160}$ respectively. Unfortunately, all these classes are of non-alternating group and cannot be shared with 3 shares, i.e., no solution exists either by decomposition or remasking[2]. We should use at least 4 shares (which is out of our focus), and its uniform sharing with 4 shares also needs to be done in at least 3 stages. Therefore, a 4-share version of TI S_0 can be realized in 6 stages.

The second one is constructed following the AES Sbox, i.e., inversion in GF(2^8), but with a different primitive polynomial and affine transformations. Based on the observations in [BB02, Rad04], inversion in one field can be transformed to another field by linear isomorphisms. Therefore, S_1 and the AES Sbox are affine equivalent and all difficulties to realize the AES TI Sbox hold true for S_1.

9.3.2 Crypton V0.5

Crypton V0.5 utilizes two 8-bit Sboxes, S_0 and S_1, in a 3-round Feistel, as shown in Figure 9.3(b). By swapping P_0 and P_2 the Sbox S_0 is converted to its inverse S_1. P_1 : `AF4752E693C8D1B0` belongs to the cubic class $\mathcal{C}_{295}$. Similar to the sub functions of CLEFIA, it belongs to the non-alternating group and cannot be shared with 3 shares. In short, at least 4 shares in 3 stages should be used. Further, P_0 : `F968994C626A135F` and P_2 : `04842F8D11F72BEF` are quadratic, non-bijective functions, but that does not necessarily mean that their uniform sharing with 4 shares does not exist. We have examined this issue by applying *direct sharing* [BNN+15], and we could not find their uniform sharing with either 3 or 4 shares. In this case, remasking is a potential solution. However, due to the underlying Feistel structure of S_0 and S_1, the non-uniformity of the shared P_0 and P_2 does not affect the uniformity of the resulting Sbox as long

---

[2] Alternatively, one can apply the technique presented in [KNP13].

as the sharing of the *Sbox input* is uniform. More precisely, P_0 output is XORed with the left half of the Sbox input. If the input is uniformly shared, the input of P_1 is uniform regardless of the uniformity of the P_0 output. See [BGN$^+$14b] and [BNN$^+$15], where it is shown that $a \cdot b$ (AND gate) cannot be uniformly shared with 3 shares, but $a \cdot b + c$ (AND+XOR) can be uniform if a, b, and c are uniformly shared. Therefore, a 4-share version of TI S_0 (resp. S_1) can be realized in 5 stages.

9.3.3 Crypton V1

Crypton V1 Sboxes as shown in Figure 9.3(c) are made of two 4-bit bijections P_0 : FEA1B58D9327064C, P_1 : BAD78E05F634192C and their inverse in addition to a linear layer in between. P_0 and its inverse P_0^{-1} belong to the cubic class $\mathcal{C}_{278}$, which can be uniformly shared with 3 and 4 shares but in 3 stages. Both P_1 and its inverse P_1^{-1} are affine equivalent to the non-alternating cubic class $\mathcal{C}_{295}$, that – as given above – must be shared at least with 4 shares. Therefore, in order to share each Crypton V1 Sbox, 4 shares in a construction with 6 stages should be used.

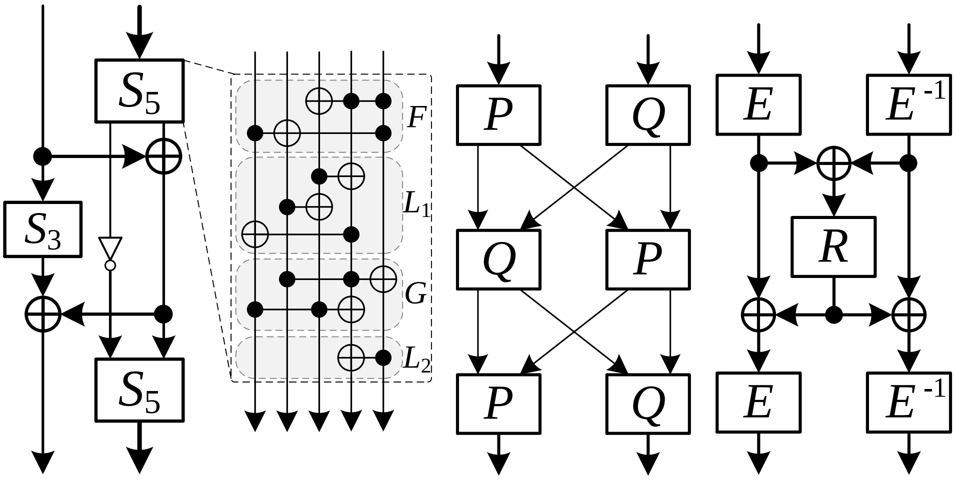
9.3.4 ICEBERG

The Sbox of ICEBERG as shown in Figure 9.3(d) is formed by two 4-bit bijections S_0 : D7329AC1F45E60B8 and S_1 : 4AFC0D9BE6173582 in a 3-round SPN structure, where permutation P_8 is a bit permutation. Both S_0 and S_1 are affine equivalent to the cubic class $\mathcal{C}_{270}$, which needs at least 3 stages to be uniformly shared with 3 shares. Therefore, a uniform sharing of the ICEBERG Sbox with 3 shares can be realized in 9 stages without any fresh randomness. Among the smallest decompositions, we suggest $A_4 \circ \mathcal{Q}_{294} \circ A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ for S_0 with A_1 : B038F47CD65E921A, A_2 : C6824E0AD7935F1B, A_3 : 3DB50E8679F14AC2, A_4 : AC24E860BD35F971, and for S_1 with A_1 : 63EB50D827AF149C, A_2 : D159F37BC048E26A, A_3 : 2AE608C43BF719D5, A_4 : C5814D09E7A36F2B, and $\mathcal{Q}_{294}$: 0123456789BAEFDC.

9.3.5 Fantomas

As shown in Figure 9.4(a) Fantomas utilizes one 3-bit bijection S_3 : 03615427 and one 5-bit bijection S_5 : 00, 03, 12, 07, 14, 17, 04, 11, 0C, 0F, 1F, 0B, 19, 1A, 08, 1C, 10, 1D, 02, 1B, 06, 0A, 16, 0E, 1E, 13, 0D, 15, 09, 05, 18, 01 in a 3-round MISTY construction. S_3 is affine equivalent to the quadratic class $\mathcal{Q}_3^3$, which can be uniformly shared with 3 shares in at least 2 stages. As a decomposition, we considered $S_3 : A_3 \circ \mathcal{Q}_1 \circ A_2 \circ \mathcal{Q}_2 \circ A_1$ with A_1 : 07342516, A_2 : 02461357, A_3 : 01235476, $\mathcal{Q}_1$: 01234576, and $\mathcal{Q}_2$: 01234675.

The construction of S_5, as shown here, consists of 4 Toffoli gates and 4 XORs. The quadratic F and G, as well as linear parts L_1 and L_2 are correspondingly marked. Hence, we can decompose $S_5 : L_2 \circ G \circ L_1 \circ F$. The uniform sharing of both F and G can be found by direct sharing. Therefore, the Fantomas Sbox can be uniformly shared with 3 shares in 4 stages, without any fresh mask. Figure 9.5(a) depicts the block diagram representation, and the area requirements are listed in Table 9.2. Each Sbox cannot be implemented iteratively, and each Sbox computation has a latency of 4 clock cycles. However, a pipeline design can send out Sbox results in consecutive clock cycles, but with a 4-clock-cycle latency.



(a) Fantomas (b) Khazad (c) Whirlpool

Figure 9.4: Structure of the Sboxes of Fantomas, Khazad, and Whirlpool.

9.3.6 Khazad

Khazad utilizes the Anubis Sbox, which is also based on a 3-round SPN as shown in Figure 9.4(b). Two 4-bit bijections P : `3FE054BCDA967821` and Q : `9E56A23CF04D7B18` in addition to a bit permutation layer form the 8-bit Sbox. Similar to ICEBERG, both P and Q belong to the cubic class $\mathcal{C}_{270}$. Therefore, the uniform sharing of the Khazad (resp. Anubis) Sbox can be realized in 9 stages without fresh masks. For the decomposition, we suggest $A_4 \circ \mathcal{Q}_{294} \circ A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ for P with A_1 : `04C862AE15D973BF`, A_2 : `A2E680C4B3F791D5`, A_3 : `842EA60CB71D953F`, A_4 : `80D5C491A2F7E6B3`, and for Q with A_1 : `082A3B194C6E7F5D`, A_2 : `3FB71D952EA60C84`, A_3 : `19D53BF708C42AE6`, A_4 : `0B38291A4F7C6D5E`.

9.3.7 Robin

Robin is constructed based on a 3-round Feistel-network similar to Crypton V0.5, but a single 4-bit bijection S_4 plays the role of all functions P_1, P_2, and P_3. Although the swap of the nibbles in the last Feistel round is omitted, the Robin Sbox is the only known 8-bit Sbox which can be implemented in an iterative fashion. S_4 : `086D5F7C4E2391BA` has been taken from [UCI+11], known as the Class 13 Sbox. S_4 is affine equivalent to the cubic class $\mathcal{C}_{223}$ and, as stated above, can be uniformly shared with 3 shares in 2 stages. As one of the smallest solutions we considered $A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ with A_1 : `AE268C04BF379D15`, A_2 : `C480A2E6D591B3F7`, A_3 : `20A8B93164ECFD75`. Therefore, with no extra fresh randomness we can realize a uniform sharing of the Robin Sbox with 3 shares in 6 stages.

In order to implement this construction, we have four different options. A block diagram of the design is shown in Figure 9.5(b) (the registers filled by the gray color are essential for pipeline designs).

(1) Iterative, w/o pipeline, each Sbox in 6 clock cycles.

(2) Iterative, pipeline, each two Sboxes in 6 clock cycles.

(3) Raw, w/o pipeline, each Sbox in 6 clock cycles.

(4) Raw, pipeline, each 6 Sboxes in 6 clock cycles, each one with a latency of 6 clock cycles.

Note that extra control logic (such as multiplexers) is required for all iterative designs which is excluded from Figure 9.5(b) and Table 9.2 for the sake of clarity.

9.3.8 Scream V3

Scream V3 is similar to that of Crypton V0.5, i.e., 3-round Feistel. P_0, and P_2 are replaced by two *almost perfect nonlinear* (APN) functions $APN1$: `020B300A1E06A452` and $APN2$: `20B003A0E1604A25`, and P_1 by S_1 : `02C75FD64E8931BA`. Similar to Crypton V0.5, the two APN functions are not bijective. However, they are cubic rather than quadratic. The source of these two APNs is the construction given in [CDL15]. We can decompose both of them into two quadratic functions as $APN1: \ F \circ G$ and $APN2: \ F \circ (\oplus 1) \circ G$, with F : `020B30A01E06A425` and G : `0123457689ABCDFE`. By $(\oplus 1)$ we represent an identity followed by XOR with constant `1`, i.e., flipping the least significant bit. Uniform sharing of G with 3 shares can be easily achieved by direct sharing. F, however, cannot be easily shared. F consists of three 2-input AND gates which directly give three output bits. To the best of our knowledge, F cannot be uniformly shared without applying remasking. However, as stated for Crypton V0.5, the non-uniformity of F (in general $APN1$ and $APN2$) does not play any role if S_1 is uniformly shared.

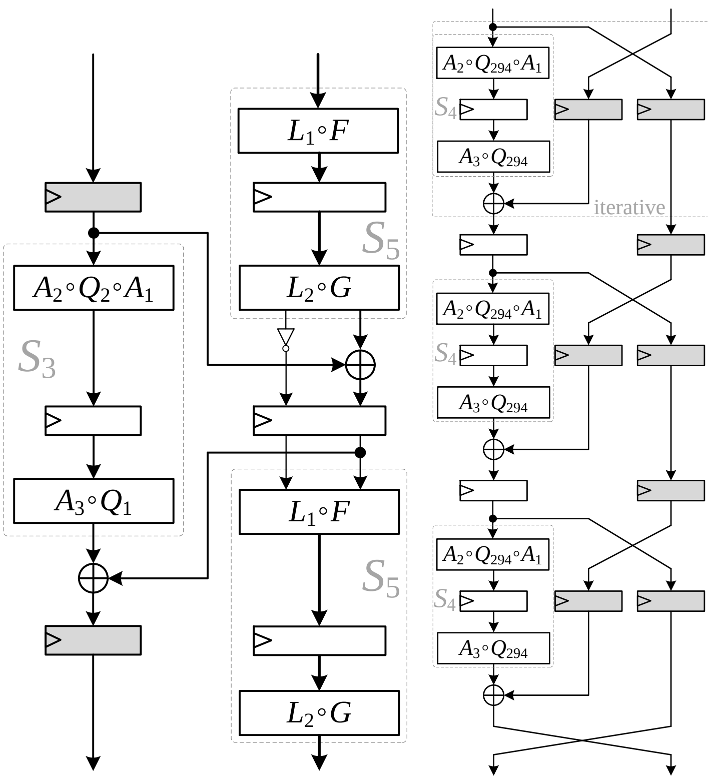
S_1 is affine equivalent to the cubic class $\mathcal{C}_{223}$ which can be uniformly shared in 2 stages with 3 shares. Therefore, the Scream V3 Sbox can be shared by 3 shares in 6 stages, without any fresh random masks. There are many options to decompose S_1; as one of the smallest solutions we suggest $S_1: \ A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ with A_1 : `26AE159D37BF048C`, A_2 : `4C086E2A5D197F3B`, A_3 : `082A3B194C6E7F5D`.

9.3.9 Whirlpool

Whirlpool employs three different 4-bit bijections E, E^{-1} and R in a customized SPN depicted in Figure 9.4(c). E : `1B9CD6F3E874A250` and its inverse are affine equivalent to the cubic class $\mathcal{C}_{278}$, which can be uniformly shared with 3 shares in at least 3 stages. R : `7CBDE49F638A2510` also belongs to the cubic class $\mathcal{C}_{270}$. As given for ICEBERG and Khazad, $\mathcal{C}_{270}$ needs 3 stages for a uniform sharing with 3 shares. Hence, the entire Whirlpool Sbox can be uniformly shared with 3 shares in 9 stages, without any extra randomness. The decomposition of R is similar to that of Khazad, i.e., $R : A_4 \circ \mathcal{Q}_{294} \circ A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ with A_1 : `02138A9BCEDF4657`, A_2 : `0C48A6E21D59B7F3`, A_3 : `C509E72BD418F63A`, A_4 : `0A1B4E5F28396C7D`. However, the decomposition of E and E^{-1} are more costly. One of the cheapest solutions is $A_4 \circ \mathcal{Q}_{294} \circ A_3 \circ \mathcal{Q}_{293} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ for E with A_1 : `048CAE2673FBD951`, A_2 : `80C4B3F7A2E691D5`, A_3 : `0B834FC71A925ED6`, A_4 : `014589CD2367ABEF`, and for E^{-1} with A_1 : `A2F76E3B80D54C19`, A_2 : `A280E6C4B391F7D5`, A_3 : `95F31D7B84E20C6A`, A_4 : `2736AFBE05148D9C`, and $\mathcal{Q}_{293}$: `0123457689CDEFBA`.

9.3.10 Implementation

We have implemented TI for all the aforementioned Sboxes except for CLEFIA, Crypton V0.5, and Crypton V1 (since they require a minimum of four shares) and have given their area requirements as well as the number of stages (clock cycles) in Table 9.2. For the synthesis, we



(a) Fantomas (b) Robin

Figure 9.5: Threshold Implementation of Robin and Fantomas Sboxes, each signal represents 3 shares, the gray registers for pipeline variants.

used Synopsys Design Compiler with the UMCL18G212T3 [Vir04] ASIC standard cell library, i.e., UMC 0.18μm technology node. It is noteworthy that among all the Sboxes we covered, the Robin Sbox is the only one which can be iteratively implemented. We should also emphasize that Midori [BBI$^+$15] and Skinny [BJK$^+$16] (in their 128-bit versions) make use of 8-bit Sboxes. Midori 8-bit Sboxes are made by concatenating two 4-bit Sboxes and the Skinny one by four times iterating an 8-bit quadratic bijection. In both cases their differential and linear properties are 64 and 128 respectively, which are notably less compared to the strong 8-bit Sboxes listed in Table 9.2. Therefore, we did not consider them in our investigations.

9.4 Finding TI-Compliant 8-Bit Sboxes

Our goal is to find strong 8-bit Sboxes which can be efficiently implemented as threshold implementations. To this end, we incorporate the idea of building an 8-bit Sbox from smaller Sboxes in our search. In particular, we aim to construct a round function that can be easily shared and iterated to generate a cryptographically strong Sbox. Easily shareable in our context refers to functions for which an efficient uniform shared representation is known. Thus, if we find a function with these properties, the resulting sequence of round functions will be a

Table 9.2: Criteria for the 8-bit Sboxes.

	Diff.	Lin.	Deg.	Iter.	AND	**Unprotected**			**Threshold Implementation**[a]					Type
						Area[GE]		Delay	Area[GE]		Delay	Stage	Mask	
				#[b]	#[c]	itera.[d]	raw[e]	ns	itera.[d]	raw[f]	ns	#[g]	#[h]	
AES [DR02]	**4**	**32**	**7**		32[BP10b]		236	5.69		4244[MPL+11]		5	48	Inversion
										3708[BGN+15]		3	44	
										3653[BGN+15]		3	44	
										2835[BGN+15]		3	32	
CLEFIA (S_0) [SSA+07]	10	56	**7**							4 shares		6	0	SPN
CLEFIA (S_1) [SSA+07]	**4**	**32**	**7**							like AES		3	32	Inversion
Crypton V0.5 [Lim98]	**8**	64	5				68	1.76		4 shares		5	0	Feistel
Crypton V1 [Lim99]	10	64	6				111	2.40		4 shares		6	0	SPN
ICEBERG [SPR+04]	**8**	64	**7**				151	2.39		2115	1.67	9	0	SPN
Fantomas [GLSV14]	16	64	5		**11**		130	2.43		766	1.72	4	0	MISTY
Khazad [BR00]	**8**	64	**7**				154	2.48		2062	1.87	9	0	SPN
Robin [GLSV14]	16	64	6	3	**12**	28	79	2.37	319	1180	1.73	6	0	Feistel
Scream v3 [GLS+]	**8**	64	6		**12**		87	2.38		2204	2.00	6	0	Feistel
Whirlpool [RB01]	**8**	56	**7**				146	2.37		2203	2.08	9	0	SPN
SB$_1$	16	64	6	8	16	**8**	**57**	**1.38**	**51**	1189	**1.09**	8	0	SPN (BitP)
SB$_2$	16	64	4	**2**	**12**	46	99	1.99	253	**631**	1.70	**2**	0	SPN (Mat)
SB$_3$	**8**	60	**7**	4	24	48	198	3.98	273	1498	2.10	4	0	SPN (Mat)
SB$_4$	**8**	**56**	**7**	5	30	29	140	4.09	202	1507	2.10	5	0	Feistel
SB$_5$	10	60	**7**	9	27	12	95	3.19	78	1583	1.10	9	0	SPN (BitP)
SB$_6$	10	60	**7**	4	20	49	174	4.78	226	1247	1.95	4	0	SPN (Mat)

---

[a] with 3 shares

[b] number of iterations of a unique function

[c] number of AND gates, important for masked bit-sliced software implementations

[d] excluding the required extra logic, e.g, multiplexers and registers

[e] fully combinatorial

[f] including pipeline registers

[g] number of stages in the pipeline

[h] number of fresh mask bits required for each full Sbox

good cryptographic Sbox which can be efficiently masked. As done previously, we concentrate on the three basic constructions mentioned above: Feistel, SPN, and MISTY. As the number of possible choices for SPN is too large for an exhaustive search, we focus on two special cases for the linear layer of the SP-network. First, instead of allowing general linear layers we focus on bit-permutations only. Those have the additional advantage of being basically for free, both in hardware and in a (bitsliced) software implementation. Second, we focus on linear layers which correspond to matrix multiplications over $\mathbb{F}_{16}$. Those cover the MISTY construction as a special case.

In all cases, the building blocks for our round function are 4-bit Sboxes. As described in Section 9.2, those Sboxes are well-analyzed and understood regarding both their threshold implementation [BNN+15] and their cryptographic properties. To minimize the number of required shares, we mainly consider functions with a maximum degree of two. Additional shares, otherwise, may increase the area or randomness requirements for the whole circuit. In [BNN+15], six main quadratic permutation classes are identified which are listed in Table 9.1. All existing quadratic 4-bit permutations are affine equivalent to one of those six. However, it should be noted that permutations of class $\mathcal{Q}^4_{300}$ cannot be easily shared with three shares without decomposition or additional randomness. Therefore, we mainly focus on the other classes from our search. Note that we include the identity function $\mathcal{A}^4_0$ in the case of the SPN construction. Since the identity function does not require any area, round functions based on a combination of identity and one quadratic 4-bit permutation can result in very lightweight designs.

One important difference to all previous constructions listed in Table 9.2 is that we do consider higher number of iterations for our constructions. This is motivated by two observations. First, it might allow improving upon the cryptographic criteria and second it might be beneficial to actually use a simpler round function, in particular those that can be implemented in one stage, more often than a more complicated round function with a smaller number of iterations. As can be seen in Table 9.2, this approach of increasing the number of iterations is quite successful in many cases.

Next we describe in detail the search for good Sboxes for each of the three constructions we considered.

9.4.1 Feistel-Construction

As a first construction, we examine round functions using a Feistel-network similar to Figure 9.1(a). By the basic approach described below, we were able to exhaustively investigate all possible constructions based on any 4-bit to 4-bit function for any number of iterations between 1 and 5. This can be seen as an extension (in the case of $n = 4$ and for identical round functions) to the results given in [CDL15] where up to 3 rounds have been studied.

However, such an exhaustive search is not possible in a naive way. As there are 2^{64} 4-bit functions and checking the cryptographic criteria of an n-bit Sbox requires roughly 2^{2n} basic operations, a naive approach would need more than 2^{80} operations.

Fortunately, this task can be accelerated by exploiting the distinct structure of Feistel-networks while still covering the entire search space.

We recall the definition of a Feistel round for the function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$:

$$\text{Feistel}_F^1 : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n \times \mathbb{F}_2^n, \qquad (L, R) \mapsto (R \oplus F(L), L).$$

We denote by Feistel_F^n the nth functional power of Feistel_F^1, i.e.,

$$\text{Feistel}_F^n = \text{Feistel}_F^1 \circ \text{Feistel}_F^1 \circ \cdots \circ \text{Feistel}_F^1 .$$

To reduce the search space, we show below that if $G = A \circ F \circ A^{-1}$ for an invertible affine function A, then Feistel_F^n is affine equivalent to Feistel_G^n.

Thus, we can reduce our search space from all 2^{64} functions, to roughly $2^{46.50}$ functions. Indeed, Brinkmann classified all 4 to 4 bit functions up to extended affine equivalence [Bri08]. There are 4713 equivalence classes up to extended affine equivalence. The following proposition summarizes the equivalence we described above.

Proposition 9.4.1. Let F and G be such that there exists an affine function A such that $G = A \circ F \circ A^{-1}$. We define:

$$\begin{aligned} B : \mathbb{F}_2^n \times \mathbb{F}_2^n &\to \mathbb{F}_2^n \times \mathbb{F}_2^n \\ (L, R) &\mapsto (A(L), A(R)). \end{aligned}$$

Then we have $\text{Feistel}_G^1 = B \circ \text{Feistel}_F^1 \circ B^{-1}$.

Proof. $\forall (L, R) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$

$$\begin{aligned}
\text{Feistel}_G^1(L, R) &= (R \oplus G(L), L) \\
&= (R \oplus A(F(A^{-1}(L))), L) \\
&= (A(A^{-1}(R) \oplus F(A^{-1}(L)), A(A^{-1}(L))) \\
&= B(A^{-1}(R) \oplus F(A^{-1}(L)), A^{-1}(L)) \\
&= B(\text{Feistel}_F^1(A^{-1}(L), A^{-1}(R))) \\
&= B(\text{Feistel}_F^1(B^{-1}(L, R)))
\end{aligned}$$

□

It follows that $\text{Feistel}_G^n = B(\text{Feistel}_F^n(B^{-1}))$, since middle terms cancel each other. Thus, we have Feistel_G^n is affine equivalent to Feistel_F^n, and have the same cryptanalytic properties. In Figure 9.6 we represent the two equivalent representation of Feistel_G^1.

Now, it is enough to consider all functions of the form $A_1 \circ F + C$, where A_1 is an affine permutation and C is any linear mapping on 4 bits. As $\text{Feistel}^n_{A_1 \circ F \circ A_2 + C'}$ is affine equivalent to the function $\text{Feistel}^n_{A_2 \circ A_1 \circ F \circ A_2 \circ A_2^{-1} + C' \circ A_2^{-1}} = \text{Feistel}^n_{A_2 \circ A_1 \circ F + C}$, this will exhaust all possibilities up to affine equivalence. Doing so, we reduce the search space to:

$$\#\mathit{Sboxes} = 4713 \cdot 2^4 \cdot |\mathrm{GL}(2,4)| \cdot 2^{16} \simeq 2^{46.50}. \tag{9.1}$$

As this is still a large search space, we employed GPUs to tackle this task.

9.4.2 SPN-Construction with Bit-Permutations as the Linear Layer

In addition to Feistel-networks, we examined round functions which are similar to Figure 9.1(c). However, $\mathcal{A}$ is replaced by an XOR with a constant followed by an 8-bit permutation. Depending on F_1 and F_2, this construction can lead to very lightweight round functions since constant addition and simple bit permutations are very efficient in hardware circuits. For F_1 and F_2 we consider the five quadratic permutations (listed in Table 9.1) as well as the identity function (denoted by $\mathcal{A}_0^4$). Obviously, we exclude the combination $F_1 = F_2 = \mathcal{A}_0^4$. There are 8! different 8-bit permutations and 256 possibilities for the constant addition. If we looked for all combinations of all affine equivalents of the chosen functions, we would have to test

$$\#\mathit{Sboxes} = 256 \cdot 8! \cdot 35 \cdot 322560^4 \cdot 10 \simeq 2^{105} \tag{9.2}$$

Sboxes. This is clearly not feasible. Therefore, we decide to restrict the number of possibilities for each of the two functions. In particular, we only consider the representative for each class as presented in [BNN+15] without affine equivalents. This reduces the search space to

$$\#\mathit{Sboxes} = 256 \cdot 8! \cdot 35 \cdot 10 \simeq 2^{32}, \tag{9.3}$$

which can be completely processed.

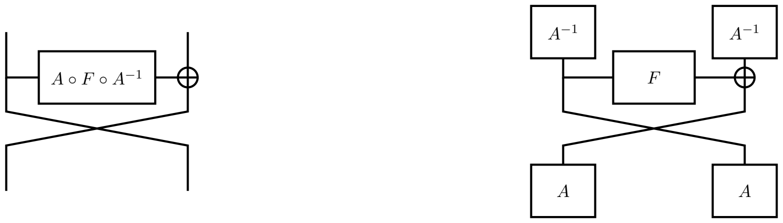


Figure 9.6: Illustration of the two equivalence representation of Feistel^1_G.

Similar to the Feistel-network, it is possible to further reduce the complexity of the search. To this end, we first define the round function for this type of Sbox as

$$\begin{aligned}\text{BitPerm}^1_{F_1,F_2,C,P} : \mathbb{F}_2^n \times \mathbb{F}_2^n &\to \mathbb{F}_2^{2n}\\ (L,R) &\mapsto P\big((F_1(L)||F_2(R)) \oplus C\big),\end{aligned}$$

instantiations where || denotes the concatenation of the two parts. Furthermore, it can be trivially seen that for every combination of an 8-bit permutation P_1 and an 8-bit constant C_1 there exist a complementary combination of an 8-bit permutation P_2 and an 8-bit constant C_2 with

$$P_1\big((L||R) \oplus C_1\big) = P_2\big((R||L) \oplus C_2\big), \ \ \forall\ R, L \in \mathbb{F}_2^n.$$

Thus, the search can be speed up since $\text{BitPerm}^1_{F_1,F_2,C_1,P_1}$ is the same as $\text{BitPerm}^1_{F_2,F_1,C_2,P_2}$. Therefore, we only need to check

$$\#Sboxes = 256 \cdot 8! \cdot 20 \cdot 10 \simeq 2^{31} \tag{9.4}$$

Sboxes for this type of round function.

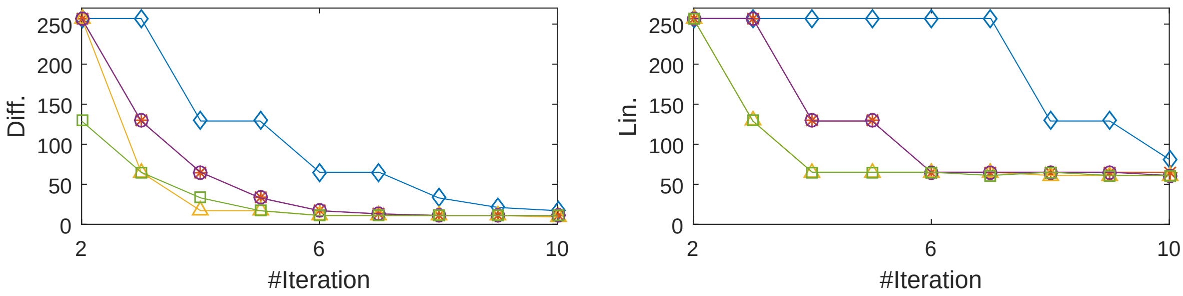
9.4.3 SPN-Construction with $\mathbb{F}_{16}$-linear Layers

For the last type, we consider another special case of the construction depicted in Figure 9.1(c). Here we restrict ourselves to the case where $\mathcal{A}$ corresponds to a multiplication with a 2×2 matrix with elements from $\mathbb{F}_{16}$. Additionally, a constant is again added to the outputs of F_1 and F_2. As noted before, a special case of this construction is the MISTY technique.

For F_1 and F_2 we consider the five quadratic functions and the identity function. Just like for the bit permutation round function, it is not feasible to check all affine equivalents. Therefore, we limit our search to these functions. The field multiplication is performed with the commonly used polynomial $X^4 + X + 1$ [GPPR11]. Given that the matrix needs to be invertible and provide some form of mixture between the two halves, this leaves us with 61200 possibilities for the matrix multiplication. It is further possible to apply the same optimization as for permutation-based round functions. Therefore, we need to check

$$\#Sboxes = 256 \cdot 61200 \cdot 20 \cdot 10 \simeq 2^{31.5} \tag{9.5}$$

Sboxes for this type of round function.



(a) Differential Uniformity

(b) Linearity

Figure 9.7: The smallest achievable differential uniformity and linearity for each number of iterations for round functions with $\mathbb{F}_{16}$-linear layers and $F_1 = \mathcal{A}_0^4$ and $(\diamond) F_2 = \mathcal{Q}_4^4$, $(*) F_2 = \mathcal{Q}_{12}^4$, $(\triangle) F_2 = \mathcal{Q}_{293}^4$, $(\circ) F_2 = \mathcal{Q}_{294}^4$, $(\square) F_2 = \mathcal{Q}_{299}^4$.

9.5 Results

We completed the search for the three aforementioned types of round functions with up to ten iterations. The search for Feistel-networks for all 4713 classes takes around two weeks on a machine with four NVIDIA K80s for a specific set of parameters. In particular, the performance depends on the bounds defined by cryptographic properties (differential uniformity) as well as the iteration count of the network. Note that, with respect to cryptographic criteria, our search shows that for iterations ≤ 5 *no 8-bit balanced Feistel with identical round functions* can have a linearity below 56 *and* a differential uniformity below 8.

Furthermore, the search for SPNs with bit permutations (resp. with $\mathbb{F}_{16}$-linear layer) required around 48 hours (resp. 54 hours) on one Intel Xeon CPU with 12 cores. It was possible to detect some very basic relations between the security, number of iterations and area of the Sbox. Figure 9.7 shows the smallest differential uniformity and linearity values which can be achieved for a specific number of iterations using a round function based on the $\mathbb{F}_{16}$-linear layer with constant addition. As expected, the more iterations are applied, the higher resistance against linear and differential cryptanalysis could be achieved. The size of each of the considered quadratic permutations is given in Table 9.1. Bigger functions like $\mathcal{Q}_{293}^4$ and $\mathcal{Q}_{299}^4$ achieve good cryptographic properties with fewer iterations than smaller functions like $\mathcal{Q}_4^4$. For the other combinations of (F_1, F_2) and types of round functions the graphs behave similarly. Depending on the remaining layers of the cipher and the targeted use case, a designer needs to find a good balance between the parameters. In the following, we present a few selected Sboxes optimized for different types of applications.

In our evaluation, we only consider Sboxes with differential uniformity at most 16 and linearity of at most 64. These are the worst properties between the observed constructed 8-bit Sboxes in Table 9.2. From the cryptographic standpoint, our Sboxes should not be inferior to these functions. We identified the following strong Sboxes that cover the most important scenarios.

(1) $\mathbf{SB}_1$: This Sbox possesses a very small round function. In a serial design the round function is usually implemented only once to save area.

(2) **SB**$_2$: This Sbox is selected to enable an efficient implementation in a round-based design. For this not only the size of the round function is important but also the number of iterations. Additional iterations require additional instantiations of the round function with a dedicated register stage. Furthermore, this Sbox requires the least number of iterations and can be implemented with a very low number of AND gates. Thus, it is also suited to masked software implementations.

(3) **SB**$_3$: This Sbox has very good cryptographic properties and requires one less iteration than **SB**$_4$.

(4) **SB**$_4$: This Sbox has very good cryptographic properties.

(5) **SB**$_5$: This Sbox is similar to **SB**$_1$ which has a small round function. However, it trades area for better cryptographic properties.

(6) **SB**$_6$: This Sbox is similar to **SB**$_2$ that is optimized for raw implementations. However, it trades area for better cryptographic properties.

9.5.1 Selected Sboxes

In this section, we supply the necessary information to implement the selected Sboxes. For this, we first recall the basic structure of the round functions. Table 9.2 shows that our selected round functions consists of bit permutations and $\mathbb{F}_{16}$-linear layers. The structure of both types is similar to Figure 9.1(c). We denote the most (resp. least) significant four bits as L (resp. R). The round function $Round : \mathbb{F}_2^4 \times \mathbb{F}_2^4 \mapsto \mathbb{F}_2^8$ is then defined as

$$\text{Round}(L, R) = P\big((F_1(L)||F_2(R)) \oplus C\big),$$

where C is an 8-bit constant and $P(.)$ denotes either an 8-bit permutation or an $\mathbb{F}_{16}$-linear layer. In Table 9.3, we describe a specific bit permutation with an eight-element vector where each element denotes the new bit position, e.g., no permutation is `01234567` whereas complete reversal is `76543210`. The $\mathbb{F}_{16}$-linear layer is realized as a multiplication with a 2×2 matrix with elements in $\mathbb{F}_{16}$. Let us denote the most (resp. least) significant four input bits to the matrix multiplication as L_M (resp. R_M). The multiplication is then defined as

$$\text{MatMul}(L_M, R_M) = (E_1 \cdot L_M \oplus E_2 \cdot R_M || E_3 \cdot L_M \oplus E_4 \cdot R_M)\,,$$

where $E_1, E_2, E_3, E_4 \in \mathbb{F}_{16}$ are the elements of the chosen matrix. To describe the linear layers of our Sboxes we give the specific $[E_1, E_2, E_3, E_4]$ for each matrix in Table 9.3.

These parameters combined with the number of iterations enable the realizations of each Sbox. To increase efficiency of the TI the constant is added to only one of the shares. In some cases, the area of the design can be reduced by adding a particular constant to the two remaining shares. This is based on the fact that an additional NOT gate can turn e.g., an AND gate to a smaller NAND gate [Pos09]. The following linear layer still needs to be applied to all shares. Table 9.3 contains this condensed description of the selected Sboxes. Further, details for each of them can be found in 14.2.

For **SB**$_4$, since it uses a Feistel-network, we construct the Sbox using the round function $H(x) = G(F(x)) \oplus A(x)$, where F is taken from the 4713 equivalence classes; G and A represent the linear and affine parts respectively. H, F, G and A are all 4-bit to 4-bit functions. The full definition of the round is then simply $(L, R) \mapsto (R \oplus H(L), L)$.

Table 9.3: Specifics of the selected Sboxes.

	F_1	F_2	Const. (Hex)	Parameter	Type	Iterations
$\mathbf{SB}_1$	$\mathcal{A}_0^4$	$\mathcal{Q}_{294}^4$	04	62750413	Perm.	8
$\mathbf{SB}_2$	$\mathcal{Q}_{293}^4$	$\mathcal{Q}_{293}^4$	EE	$[2, 4, 4, 2]$	Matrix	2
$\mathbf{SB}_3$	$\mathcal{Q}_{293}^4$	$\mathcal{Q}_{299}^4$	6C	$[2, 2, 3, 11]$	Matrix	4
$\mathbf{SB}_5$	$\mathcal{Q}_4^4$	$\mathcal{Q}_{294}^4$	85	20647135	Perm.	9
$\mathbf{SB}_6$	$\mathcal{Q}_{293}^4$	$\mathcal{Q}_{294}^4$	F8	$[0, 5, 13, 15]$	Matrix	4

	F	G	A	Type	Iterations
$\mathbf{SB}_4$	0001024704638EAD	028A9B1346CEDF57	6273627351405140	Feistel	5

9.5.2 Comparison

Table 9.2 gives an overview of our results and we summarize the most important observations in the following. The first observation is that our proposed designs do not require fresh mask bits to achieve uniformity. This is an improvement over all TI types of the AES Sbox and some other Sboxes from Table 9.2. They need up to 64 bits of randomness for one full Sbox. Given that modern ciphers usually include multiple rounds with many Sboxes, this can add up to a significant amount of randomness which needs to be generated.

Furthermore, all of our proposed Sboxes can be implemented iteratively. This comes with the advantage that even the more complex designs, e.g., $\mathbf{SB}_4$ and $\mathbf{SB}_5$, can be realized with very few gates depending on the design architecture. From all the other Sboxes in Table 9.2 this is only possible for Robin and its round function requires more area than any of our proposed Sboxes.

In particular, $\mathbf{SB}_1$ and $\mathbf{SB}_2$ require the least area in their respective target architectures (i.e., iterative and raw) out of all considered 8-bit Sboxes . The difference for the iterative architecture is especially large where $\mathbf{SB}_1$ needs roughly six times less area than the Robin Sbox.

$\mathbf{SB}_2$ requires the least number of stages. Additionally, it requires only 12 AND gates for the whole Sbox which is very close to the best number, i.e., 11 for Fantomas. This is an advantage for masked bit-sliced implementations making $\mathbf{SB}_2$ suitable for software and hardware designs. For completeness, we also look at the masked bitsliced implementation of Sboxes with a low number of AND gates (≤ 16), i.e. $\mathbf{SB}_1$ and $\mathbf{SB}_2$. Software implementations are not vulnerable to glitches hence the probing model [ISW03] is good to model the security of these implementations. We use the solution for secure AND proposed in [ISW03] and take advantage of the proof of Rivain and Prouff [RP10] to limit the number of shares. The results are plotted in Figure 9.8. As expected the number of AND is determinant for large masking order and the cost of the linear part becomes negligible. In particular, $\mathbf{SB}_2$, Scream v3 and Robin have the same number of AND (12) and differ just by the linear part. The 3 curves converge toward the same curve.

As expected, we did not find any Sbox with better cryptographic properties than the AES Sbox. However, $\mathbf{SB}_3$ and $\mathbf{SB}_4$ can still provide better resistance against cryptanalysis attacks than most of the other considered Sboxes. This comes at the cost of an increased area for the

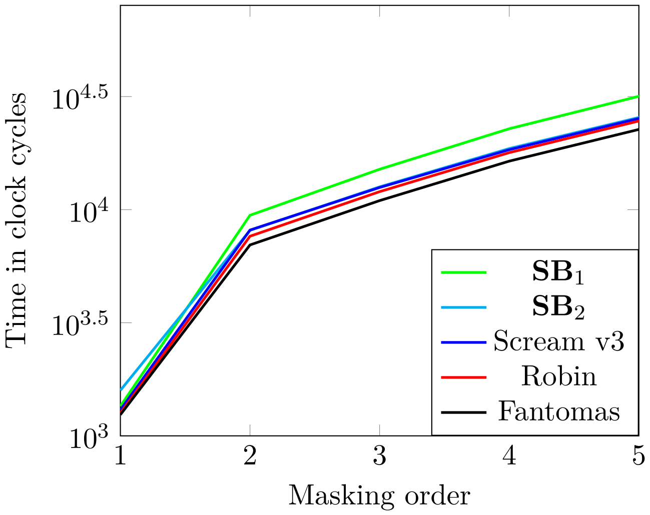


Figure 9.8: Bitslice masked implementation for the ATMEGA644p.

raw implementations. Nevertheless, the required area is still smaller than any AES TI and their round function is still smaller than Robin for iterative designs.

As depicted in Figure 9.7, a trade-off between resources and cryptographic properties is possible. If **SB**$_1$ and **SB**$_2$ do not provide the desired level of security and **SB**$_3$ and **SB**$_4$ are too large, **SB**$_5$ and **SB**$_6$ might be the best solution. Their cryptographic properties are still better or equal than the competitors while the area is significantly smaller than **SB**$_3$ and **SB**$_4$. For the sake of completeness, we included the area requirement of the unprotected implementation as well as the latency of different designs in Table 9.2.

Decryption usually requires the inverse of the Sbox. Therefore, it is important that the Sbox inverse has comparably good properties to the original Sbox. For **SB**$_4$ this is obvious since the Feistel-structure makes it straightforward to construct the inverse. Therefore, inverse **SB**$_4$ has exactly the same properties as **SB**$_4$. For the other cases, this is not trivial. Nevertheless, the inverse of each of our-considered quadratic functions is self-affine equivalent.

9.6 Conclusion and Future Work

In this work we identified a set of six 8-bit Sboxes with highly useful properties using a systematic search on a range of composite Sbox constructions. Our findings include 8-bit Sboxes that provide comparable or even higher resistance against linear and differential cryptanalysis with respect to other 8-bit Sbox but intrinsically support the TI concept without any external randomness. At the same time our selected Sboxes come with a range of useful implementation properties, such as a highly efficient serialization option, or a very low area requirement.

Future work comprises extended criteria for the Sbox composition, including diffusion layers beyond permutations and substitution layers based on other functions. It is also interesting to examine the influence on the cryptographic properties of alternating round functions in comparison to the currently fixed round function. A more general research direction related to the basic concept of TI deals with a more clear understanding of which quadratic functions can be efficiently shared for an arbitrary number of bits.

Chapter 10

ParTI: Towards Combined Hardware Countermeasures

In this chapter, we introduce a countermeasure for cryptographic hardware implementations based on [SMG16b] that combines the concept of a provably-secure masking scheme (i.e., threshold implementation) with an error detecting approach against fault injection. As a case study, we apply our generic construction to the lightweight LED cipher. Our LED instance achieves first-order resistance against side-channel attacks combined with a fault detection capability that is superior to that of simple duplication for most error distributions at an increased area demand of 12%.

Contents of this Chapter

10.1 Introduction **127**
10.2 Background **129**
10.3 Methodology **130**
10.4 Case Study: LED **142**
10.5 Conclusions and Future Work **152**

10.1 Introduction

Beside side-channel analysis, active FI attacks pose a further serious threat to instantiated cryptographic algorithms [BS97] by injecting a fault during its execution. The adversary then derives sensitive information from the erroneous output of the device. For more sophisticated attacks on symmetric schemes to work, multiple of these erroneous outputs need to be combined. Like for SCA, there are a wealth of attacks and possibilities to generate faults during the computation, e.g., by clock or power glitches or positioned photon injection using lasers. In terms of countermeasures, the majority of published concepts are based on the principle of concurrent error detection (CED). The main idea is to utilize redundancy in time or area to enable quasi-immediate detection of faults. Some CED schemes integrate the use of error detecting codes to enhance their level of protection. Over the years, various different codes have been studied to harden cryptographic implementations against FI attacks. Due to its simplicity, parity check codes are commonly used in this context [KKG03, BBK$^+$03]. Other schemes based on non-linear codes (e.g., [KKT04b, KKT04a]) were brought up due to their beneficial fault coverage. Recently, the class of infective countermeasures have been put forward which do not require an explicit final check before returning the result. [GST12].

As previously discussed, it is mandatory for cryptographic devices to integrate dedicated SCA and FI countermeasures if they are operated in untrusted environments. Still, the majority of proposed SCA and FI countermeasures have been solely evaluated separately, although both classes need to be integrated in a single device. For simple countermeasures (e.g., applying plain redundancy in area and time), a separate evaluation is justified since multiple executions of the same SCA-protected operation are admissible (with few exceptions). However, more sophisticated FI countermeasures are likely to affect the SCA countermeasure to a higher degree which can have a severe impact on the security and efficiency of the combined scheme. For example, if parity bits used by FI countermeasures are computed over *unmasked* intermediate values, it leads to a side-channel leakage even if the rest of the design in perfectly masked. Thus, a careless integration may easily lead to contradicting the assumptions of the underlying masking scheme, and hence failure of the masked design.

10.1.1 Related Work

In response, a few countermeasures have been proposed providing resistance against both kind of attacks. At the gate level, we refer to dual-rail logic styles (e.g., WDDL [TV04]) which – due to the additional presence of dual counterparts of the circuit – inherently offer a fault detection feature. However, the error detection rate is limited to the concept of simple duplication.

Furthermore, coding schemes have been used for combined countermeasures as well. Wiretap codes that have been applied as an SCA countermeasure [BCL12, Mor14b] at algorithm level, can also provide a certain level of fault detection. Additionally, there are further examples [BCC$^+$14] that use coding techniques for enhanced resistance against both types of attacks. However, most of the schemes are either designed for software implementations or provide only limited security at the expense of high overheads.

Besides combined countermeasures, there are also combined attacks which use a combination of fault injection and side-channel analysis to extract a secret. Several attacks have been proposed against protected AES implementations where masking together with various fault countermeasures are integrated [RLK11, CFGR10, DV12]. Our analyses consider this powerful threat as well.

10.1.2 Contribution

We propose a new combined physical protection scheme targeting hardware platforms. As mentioned before, the integration of CED schemes by simple *time* or *area redundancy* into masked designs is straightforward (see [XGK12] for definitions). However, such constructions are not able to detect certain types of faults (e.g., identical faults which are injected in both instances of the design) and rather costly. Therefore, our target in this work is to merge more sophisticated *information redundancy* approaches (namely *error detecting codes*) with provably secure masked hardware designs. More precisely, we demonstrate how to integrate an error detecting code into first- (or higher-) order TI designs, while preserving all security requirements and features of the underlying TI concept. We formalize our methodology to allow various types of codes which provide the most flexibility in terms of protection and area requirement. We include a thorough analysis on the resistance of the combined countermeasure regarding the chosen order of TI and the parameters of the code. Note that the straightforward hardware duplication can be regarded as a subtype of our combined countermeasure, but our generic concept enables to tweak the protection of the resulting design by the choice of code.

For practical evaluation we present a case-study on the cipher LED [GPPR11] that simplifies the explanation of the underlying concept due to its simple structure. We provide practical evaluations of our design implemented on an FPGA with respect to any detectable first-order leakage. Moreover, we evaluate the performance, area overhead as well as the fault coverage of the integrated *information redundancy* scheme. Note that the representations included in this work primarily discuss the case of a first-order TI design of LED with fault detection facility based on Hamming codes. But we like to emphasize that our generic construction can be similarly applied to any-order TI designs of other ciphers that are using different error detecting codes.

10.2 Background

We briefly introduce Error Detecting Codes (EDC) and their application in Concurrent Error Detection (CED) schemes.

10.2.1 Error Detecting Codes

EDC are primarily used to transmit data over an unreliable communication channel. Those properties and notation of EDC that are also relevant for remainder of this work will be highlighted in the following [MS77].

Definition 10.2.1. A linear code $\mathbf{C}$ of length n over $\mathbb{F}_q$ is a vector subspace over $\mathbb{F}_q^n$.

We only consider binary codes (i.e., $q = 2$) in this work since they provide the best performance for our projected use case in symmetric cryptography. A linear code $\mathbf{C}$ that maps messages of length k to codewords of length n is commonly denoted as an $[n, k]$-code.

Definition 10.2.2. A *generator matrix* G of an $[n, k]$-code $\mathbf{C}$ comprises n basis vectors of $\mathbf{C}$ with length k.

A generator matrix can be used to transform a given message $m \in \mathbb{F}_q^k$ to the corresponding code word $c \in \mathbf{C}$ as $c = m \cdot G$.

Definition 10.2.3. A matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ with the property

$$\mathbf{0} = H \cdot c^T,\ \forall c \in \mathbf{C} \tag{10.1}$$

is denoted as *parity check matrix* of the code $\mathbf{C}$.

Such matrix can be used to easily check if a given c is a valid codeword of $\mathbf{C}$.

Definition 10.2.4. The minimum distance d of a linear code $\mathbf{C}$ is defined as

$$d = min(\{wt\,(c_1 \oplus c_2)\,|c_1, c_2 \in \mathbf{C}, c_1 \neq c_2\}), \tag{10.2}$$

where $wt(x)$ returns the number of 1's in the vector x (known as Hamming weight). We denote a linear code $\mathbf{C}$ of length n, rank k and minimum distance d as an $[n, k, d]$-code. The minimum distance of a code determines its error detection and correction property.

Definition 10.2.5. A code $\mathbf{C}$ with minimum distance d can be used to either detect $u = d - 1$ or correct $v = \left\lfloor \frac{d-1}{2} \right\rfloor$ errors. If d is even, $\mathbf{C}$ can simultaneously detect $u = \frac{d}{2}$ and correct $v = \frac{d-2}{2}$ errors.

Given an erroneous codeword $c' = c \oplus e$, where e is known as the error vector, a u-error detecting code is able to detect that c' is faulty as long as $wt(e) \leq u$.

Definition 10.2.6. The generator matrix G of a systematic code $\mathbf{C}$ is of the form $G = [I_k|P]$ where I_k denotes the identity matrix of size k.

Each codeword c of a systematic code consist of the message itself which is padded by *check bits*, i.e, $c = [m|p]$. The check bits p[1] are generated by the rearward part of the generator matrix G represented by P. Note that all linear non-systematic codes can be transformed into a systematic code with the same minimum distance [Bla03].

10.2.2 Concurrent Error Detection

CED systems are commonly used to detect arbitrary faults during the execution of an operation what makes them also an appropriate countermeasure against FI attacks [GMJK15]. Typically, CED techniques rely on different types of redundancy to detect faulty computations. The most straightforward approach implements redundancy by multiple executions which results either in an increased area or in an increased time complexity. Certain intermediate values of different runs are compared with each other to detect errors.

As already indicated in the introduction, some CED schemes use error detecting codes following a structure similar to Figure 10.1 to achieve a better fault coverage. In this basic example, CED is used to protect an OPERATION which is applied to a given INPUT. Initially, the CHECKBITS of the INPUT are generated by means of the GENERATOR matrix of the code. A PREDICTOR takes INPUT and CHECKBITS and returns the predicted check bits of the output of OPERATION. These are compared with the actual CHECKBITS of the output. If a detectable error (depending on the type of the code) occurred during the execution, these two types of CHECKBITS will not be identical. Thus, a possible attack can be detected and averted. It should be noted that, depending on the target algorithm and the integrated code, the prediction functions can have an exalted level of complexity. Thus, the overhead of some CED schemes using EDC can be similar to a complete duplication of the operation.

Traditionally, the effectiveness of these fault detection countermeasures was examined in a uniform fault model. However, recent publications [GMJK15] have shown that this model does not closely resemble real-world attacks and that some of the presented countermeasures are in fact vulnerable to *biased fault attacks* [PCNM15].

10.3 Methodology

In this section, we introduce our methodology to develop a combined countermeasure against side-channel and fault injection attacks that is specifically tailored for hardware platforms. We

---

[1] Note that p can be also considered as a form of parity bits.

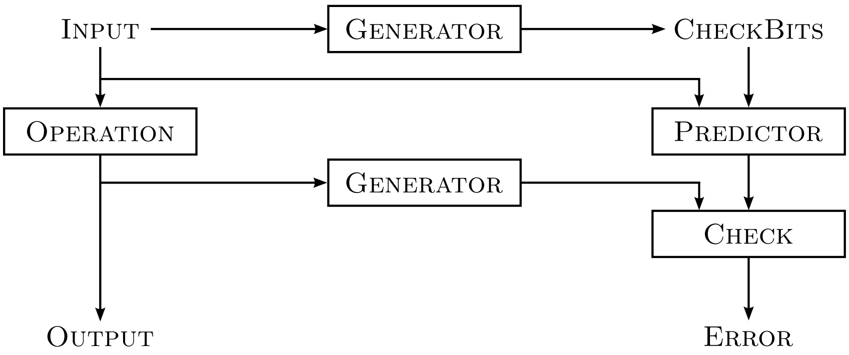


Figure 10.1: A common structure of CED schemes using EDC.

first discuss the necessary considerations and restrictions of a combined scheme. This is followed by a detailed description of the attacker model and how to design a scheme to support arbitrary applications.

10.3.1 Design Considerations

Firstly, our countermeasure is designed for hardware platforms. Thus, efficiency in software is not a concern in the design process. As hardware circuits are often used to achieve high performance, a primary design goal is to minimize the impact of the countermeasure on the performance.

Secondly, in terms of SCA countermeasure we aim at providing provable security (at least to a certain order). Therefore, hiding techniques are not applicable and we have to rely on masking. Given the first design goal, this leaves us with TI as it comes with a reasonable performance overhead compared to other masking schemes in hardware circuits [PR11].

Thirdly, our scheme aims to be more secure against (realistic) FI attacks than simple duplication. Doubling the masked hardware circuit is a straightforward way to combine masking with some form of redundancy. However, as mentioned before, simple duplication can be highly vulnerable to fault attacks if the fault model follows a different distribution than uniform. Therefore, we aim at building a scheme that is more robust against adversaries exploiting the effect of (reasonably) biased distributions. In this context, we choose EDC due to their sound theoretical foundation providing solid bounds on the number of detectable errors. Nevertheless, the balance between the error detection capability and run time performance is essential to not severely impact our first design goal.

10.3.2 Attacker Model

Since our scheme aims to provide resistance against both SCA and FI attacks, we evaluate our methodology in a model that incorporates both types of threats. In the following, we assume an adversary that can observe the physical characteristics of the design during execution and further is able to inject faults in the circuit.

We assume a computationally bounded adversary that can observe the power consumption of our design during a finite number of executions. Note that security guarantees of TI also hold with respect to other side channels, e.g., electromagnetic emanations. Due to the computational restriction of the adversary, we can bound the number of possible observations. Given that the complexity of an attack increases with its order, we bound the adversary by the highest order of an attack he is able to mount. In other words, the adversary is able to observe a limited number of executions that is just enough to perform attacks of order d but not of order $d + 1$. The actual order depends on the platform and the desired level of security.

Furthermore, the adversary is able to inject faults in the hardware circuit. In our model, we assume that injected faults only target the data path of the implementation and exclude the control flow. This is a different aspect of fault attacks which is not specific to our scheme. Given that the control flow usually does not need to be protected by masking, it is not considered in our combined countermeasure. Nevertheless, our combined countermeasure needs to be implemented together with a protected control flow to ensure complete security. There are various solutions to this problem. Even the EDC aspect from our combined countermeasure can be used to harden the control flow as described in [SGS08]. Therefore, we model the injected faults as an error random variable E following a specific distribution $\mathcal{E}$. In our model, an error vector $e \in \mathbb{F}_q^n$ with probability $Pr[E = e]$ is sampled for each injected fault from the distribution and added (XORed) to the current $state$ of the execution as $state' = state \oplus e$. The execution continues the computation with the altered state $state'$. In the following, we consider two different types of the error distribution.

Since most existing works assume a uniform fault model, we also first examine our combined countermeasures against an adversary with a uniform distribution $\mathcal{E}_U$ so that $Pr[E = e_1] = Pr[E = e_2], \forall e_1, e_2 \in E$.

Furthermore, we consider a biased distribution $\mathcal{E}_B$, where one specific set of error vectors $E_1 \subset E$ is significantly more probable than the set of remaining error vectors $E_2 \subset E$ with $Pr[E = e_1] \gg Pr[E = e_2], \forall e_1 \in E_1, e_2 \in E_2$. The sets are determined by the type of faults that are considered in the model, e.g., $E_1 : \forall e, wt(e) \leq u$. In an extreme case, E_1 only contains one specific error vector e with $wt(e) = 1$. This scenario is akin to laser-based fault injections in which single bits can be targeted.

10.3.3 Code Selection

Obviously, the choice of the code strongly affects the efficiency and fault coverage of the resulting combined scheme. In this context, it is not possible to provide one specific code that exhaustively fits to all possible application scenarios. Instead, the code needs to be specifically chosen according to the target algorithm to yield optimal results. A poorly chosen code can cause a significant overhead while offering only little benefit in terms of fault coverage. In this subsection we discuss necessary considerations made in the code selection process and give guidelines on the criteria how to pick a code.

Linear Codes

One important aspect in the design of a TI is the algebraic degree of the targeted functions. As explained in Section 7.1, the algebraic degree determines the minimum number of necessary shares. Given that the prediction functions are also part of the intended TI, it is crucial that they possess the same algebraic degree as the original function. Otherwise, the requirement of an additional share negatively affects the area complexity of the resulting design. This property is trivially fulfilled by linear codes. The encoding and decoding functions of linear codes are linear. Therefore, adding a decoding function before and an encoding function after the target function (cf. Figure 10.3) to obtain the predictor guarantees that the emerging function has the same algebraic degree as the target function. For non-linear codes this property is not always satisfied. In addition, the encoding/decoding functions of linear codes can be implemented extremely efficiently which makes the necessary error check also very efficient. In the remainder of this work, we therefore only consider linear codes.

Systematic Codes

Systematic codes are advantageous to improve the efficiency. Due to their specially structured generator matrix (cf. Definition 10.2.6), the output of the targeted function does not need to be decoded to recover the correct result since the message is part of the codeword. This helps to eliminate one otherwise necessary step at the end of the design. Furthermore, the distinction between target function and predictor – as depicted in Figure 10.1 – is otherwise not easily possible. Since one half of the design is nearly completely unaltered by the inclusion of fault countermeasure, it also allows reuse of existing TI designs. This design decision does not limit the choice of codes since (as already mentioned in Section 10.2.1) every linear non-systematic code can be transformed into a systematic code with the same minimum distance.

Code Parameters

The choice of the three parameters of a linear code n, k, and d depends on the target algorithm. A good practice is to derive the code dimension k from the size of a single element that is used in most functions of the targeted algorithm, e.g., for an algorithm that performs most of its operations in $GF(2^8)$ it is advisable to set $k = 8$. This way unnecessary overhead due to the split or merge of check bits is avoided. Furthermore, the code length n also affects both the efficiency and fault coverage of the design. To achieve a desired error detecting level of $u = d - 1$, a certain minimal size of n is required. However, if n is chosen too large, the number of check bits increases resulting in a high area complexity. Therefore, it is important to find a good trade-off between the length of the code n and its minimum distance d. In the following, we aim at a design in which the predictors work solely on the check bits. To achieve this, it is necessary that the message m can be fully recovered using the check bits p. Assuming that the message has full entropy (which is usually the case in symmetric cryptographic applications), it is advisable to set the rank to at least $n \geq 2k$.

10.3.4 Threshold Implementations with Error Detecting Codes

To achieve the desired level of security against SCA adversaries, it is necessary to implement all required functions according to the principles of TI. In particular, this includes the prediction functions as well. As it was already thoroughly discussed in [BNN$^+$12, BNN$^+$15], we omit the detailed explanation how to construct TI-compliant shared representations of arbitrary functions. Instead, we describe the specifics of including EDC in a TI design and how to easily find the TI of the predictors.

Notation

In the following, we assume a systematic linear code, which allows message recovery from the check bits. Further, we denote the input to the target algorithm by $\boldsymbol{m}_i$ with $n_m = |\boldsymbol{m}_i|$ as its bit length and the corresponding check bits as $\boldsymbol{p}_i$ with $n_p = |\boldsymbol{p}_i|$. The output of the target algorithm and its corresponding check bits are indicated by $\boldsymbol{m}_o$ and $\boldsymbol{p}_o$ respectively. Since the code does not change during the execution, the outputs have the same size as their corresponding inputs. Further, we assume that the TI of the target algorithm requires a minimum of s shares to be secure. To this end, the messages and their corresponding check bits need to be masked accordingly as

$$\boldsymbol{m}_i = \bigoplus_{j=1}^{s} \boldsymbol{m}_i^j, \qquad \boldsymbol{p}_i = \bigoplus_{j=1}^{s} \boldsymbol{p}_i^j, \qquad \boldsymbol{m}_o = \bigoplus_{j=1}^{s} \boldsymbol{m}_o^j, \qquad \boldsymbol{p}_o = \bigoplus_{j=1}^{s} \boldsymbol{p}_o^j.$$

Basic Structure

Due to the special characteristics of the chosen code, it is possible to split up the computations of the underlying target algorithm and the predictors. The two output values $\boldsymbol{m}_o$ and $\boldsymbol{p}_o$ are calculated completely independent of each other. This leads to the basic structure as depicted in Figure 10.2. There is an additional element (Error Check) which receives intermediate states of both circuits as input and checks if an error has occurred. The frequency for these checks is a variable in the specific design process, but it affects both the area and the fault coverage of the complete circuits. The higher the check frequency, the higher is the fault coverage but also the area requirements. In the most basic approach, only $\boldsymbol{m}_o$ and $\boldsymbol{p}_o$ are checked after a cipher run is complete.

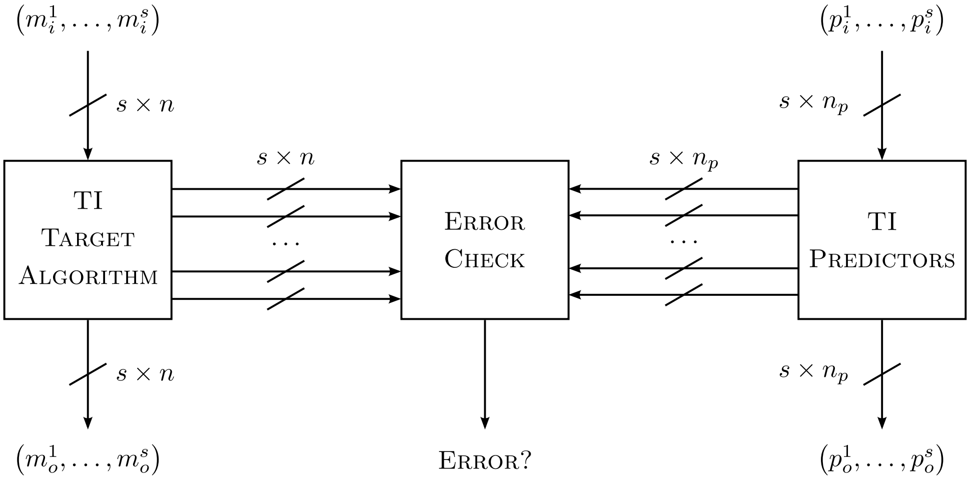


Figure 10.2: The basic structure of our combined scheme.

For some TI designs a mask refresh during the execution is necessary to retain uniformity, e.g., for the AES Sbox [BGN+14a]. Given that our proposed predictors are identical to the target function with an initial and final affine transformation, it is likely that they require a mask refresh depending on the shared function. Obviously, this can lead to a non-negligible overhead depending on the target algorithm. However, the separate computation paths (of the original and predictors) allow reusing the fresh randomness to some degree. Since both parts are completely independent and their respective intermediate values are never given to a joint function (except for the error check), it is possible to use the same random bits to refresh both sides. In the other case, where the predictors get inputs from both sides, this is not feasible without harming the uniformity property which would violate the security proofs of TI. For the error check, it is necessary to compute a function which takes inputs from both sides. However, in our scheme (and many others) this check can be implemented in a way that it only leaks the occurrence of a fault. For this to work, it is necessary that both sides use the same random masks which enables a separate error check on every share as also proposed as a countermeasure against combined attacks in [RLK11, DV12]. This has obviously an impact on the efficiency of the design since the total amount of randomness is reduced due to the mask reuse.

We illustrate this problem with an example. Let us assume a function F with two input bits a, b with $F(a,b) = ab$. The corresponding check bit is defined as $c = a + b$ with the predictor $F_p(a,c) = a + ac$. As noted in [BNN⁺15], there is no uniform sharing of F. Instead, a virtual share is added to achieve uniformity. The shared functions using one virtual share are

$$F_1 = a_2b_2 + a_2b_3 + a_3b_2 + r \tag{10.3}$$

$$F_2 = a_3b_3 + a_1b_3 + a_3b_1 + a_1r + b_1r \tag{10.4}$$

$$F_3 = a_1b_1 + a_1b_2 + a_2b_1 + a_1r + b_1r + r, \tag{10.5}$$

where r is randomly drawn from a uniform distribution. Analogously, the predictor can be shared as

$$F_{p1} = a_2 + a_2c_2 + a_2c_3 + a_3c_2 + r \tag{10.6}$$

$$F_{p2} = a_3 + a_3c_3 + a_1c_3 + a_3c_1 + c_1r \tag{10.7}$$

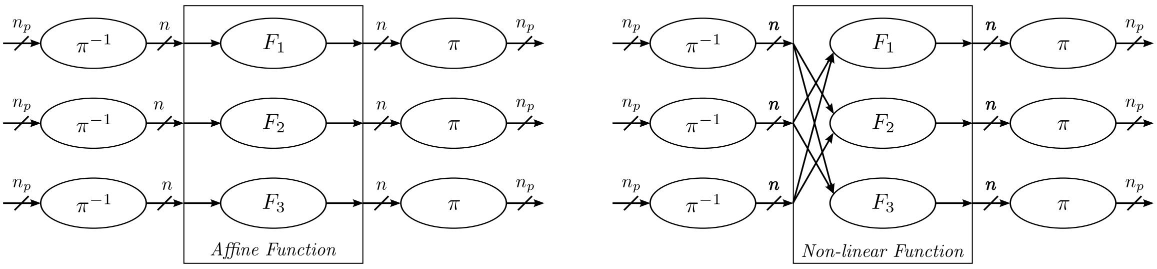
$$F_{p3} = a_1 + a_1c_1 + a_1c_2 + a_2c_1 + c_1r + r. \tag{10.8}$$

If both (F_1, F_2, F_3) and (F_{p1}, F_{p2}, F_{p3}) share the same r, the resulting six output bits would not be jointly uniform. Meaning that, they cannot be used as input to another joint function (i.e., another predictor) without violating the uniform input property of TI. To fix this, double amount of fresh randomness (i.e., one r bit for each part) is required.

Shared Predictors

Contrary to ordinary CED schemes, our predictors need to comply with the requirements of TI. In other words, the prediction functions work on masked check bits and fulfill the non-completeness, correctness, and uniformity properties. Finding functions with all these characteristics can be difficult for certain codes. However, in our presented scenario (i.e., a systematic linear code with a *sufficiently large* rank) it can be significantly simplified.

The general approach is shown in Figure 10.3 with the example of affine and non-linear functions with three shares. $\pi : \mathbb{F}_2^{n_m} \to \mathbb{F}_2^{n_p}$ denotes the generation of the check bits using P, the right part the generator matrix. Respectively, $\pi^{-1} : \mathbb{F}_2^{n_p} \to \mathbb{F}_2^{n_m}$ is defined as its inverse, i.e., recovery of the message from the check bits. To derive the shared representation of the predictor from the target functions, each input share is first transformed using π^{-1}. Then the target function is applied and each resulting share is run through π to generate the corresponding



(a) affine (b) non-linear

Figure 10.3: Derivation of shared predictors for three shares.

check bits again. Of course, the steps do not have to be performed segregated. Instead, they are merged and subsequently optimized to achieve a better performance. The resulting functions trivially comply with the correctness property. Given that π and π^{-1} operate on single shares, non-completeness is also maintained. Regarding the uniformity of the output shares, we need to differentiate between two cases. For $n_p = n_m$, the uniformity property is preserved from the target functions as noted in [SMG15a]. The encoding and decoding operations are only affine transformations which do not influence the uniformity in this setting. However, for $n_p > n_m$ this observation does not generally hold. If the steps are performed in an isolated manner, the reduction of the input shares to size n_m will come with a reduction in entropy. In result, the enlarged output shares are no longer uniform. A trivial solution would be the inclusion of a fresh random value to restore uniformity. However, this reduces the performance of the design and is therefore undesirable. A more efficient solution is to merge the three steps (i.e., π^{-1}, F, π) and eliminate the reduction of the input shares.

Depending on the operation, this optimization can be very effective. Especially if π and π^{-1} are linear over the F they can be partially canceled out. As mentioned before, functions with a high degree are often decomposed to reduce the number of shares. Usually there are multiple possibilities for decomposition with different degrees of efficiency. Depending on the scenario, these decompositions do not need to be the same for the predictors. In these cases, the final result is still the same but not necessarily the intermediate values. This enables more efficient designs while leading to some limitations in the error detection, as discussed later on.

Error Detection

As noted before, the rate of error detection inside the algorithm affects the performance, area consumption and fault coverage of the design. Frequent error checking thwarts potential optimization of the predictors what finally leads to larger circuits.

The error checking is performed similar to Figure 10.1. However, in our basic scenario (without reusing randomness) the intermediate values are split up into multiple shares via Boolean masking. To still detect if an error has occurred, a two-step approach denoted as CHECK-AND-COMBINE is required. In the first step CHECK, the parity check matrix is multiplied with each share of the codeword. Thus, the resulting error check vectors $\boldsymbol{v}_{int}^{j}$ are computed as

$$\boldsymbol{v}_{int}^{j} = H \cdot \left(\boldsymbol{c}_{int}^{j}\right)^{T} = \pi(\boldsymbol{m}_{int}^{j}) \oplus \boldsymbol{p}_{int}^{j},\ 1 \leq j \leq s. \tag{10.9}$$

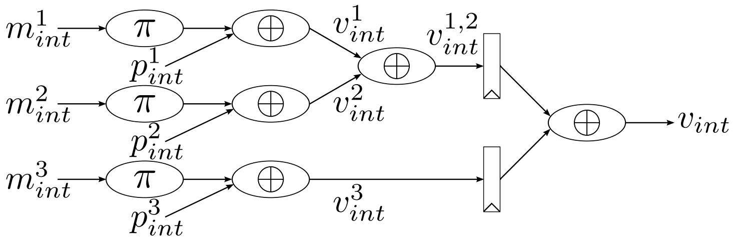


Figure 10.4: Computation and unmasking of the error check vector for three shares in a first-order secure design.

If no error has occurred, these error vectors are a random sharing of the null vector. To check this, the error vectors are combined via XOR in the second step COMBINE. However, without any registers this procedure is equivalent to a function which has all shares of both parts of the circuit as input. This certainly violates the non-completeness property of TI. To this end, it is necessary to split up the second step COMBINE into multiple parts and include registers in between. In case of a first-order secure design, all but one of the shares are first combined. The result and the last share are then stored in a register and combined as

$$\boldsymbol{v}_{int} = \left(\bigoplus_{j=1}^{s-1} \boldsymbol{v}_{int}^{j} \right) \oplus \boldsymbol{v}_{int}^{s}. \tag{10.10}$$

If $\boldsymbol{v}_{int}$ is not the null vector, an error has been detected. The last XOR technically violates the non-completeness property as it unmasks $\boldsymbol{v}_{int}$ by merging all shares. However, $\boldsymbol{v}_{int}$ holds no information about the sensitive intermediate values of the circuit. Therefore, the SCA resistance of the design is not jeopardized by this. An exemplary error check procedure with three shares is depicted in Figure 10.4. It should be noted that the initial input values are indeed in compliance with the uniformity property of TI. The input values $\boldsymbol{v}_{int}^{1,2}$ and $\boldsymbol{v}_{int}^{3}$ to the second part (right of the registers) are not jointly uniform given that if no error has occurred they are identical. Yet this does not affect the security of the resulting design since (as argued before) $\boldsymbol{v}_{int}$ does not hold any information related to sensitive intermediate values. This security guarantee still holds if the same randomness is used for masking both $\boldsymbol{m}_{int}$ and $\boldsymbol{p}_{int}$. Even though the input to the multiplication with H is not uniform, it does not pose a problem as it is applied to each share separately and the resulting $\boldsymbol{v}_{int}$ does not hold any information related to sensitive intermediate values.

The CHECK-AND-COMBINE procedure can be further simplified. To this end, it is necessary that all randomness is reused and the check bits are carefully generated and predicted during the cipher run. One possibility to generate the check bits assuming $n_p = n_m$ is

$$\boldsymbol{p}_i^j = \begin{cases} \pi\left(m_i\right) \oplus r^j, & \text{for } 1 \leq j < s \\ \pi\left(m_i\right) \oplus \left(\bigoplus_{j=1}^{s-1} r^j \right), & \text{for } j = s \end{cases} \tag{10.11}$$

where r^j denotes uniformly distributed fresh random masks with $r^j \in_R \mathbb{F}_2^k$, $1 \leq j < s$. Hence, the same masks are used for $\boldsymbol{m}_i$ and $\boldsymbol{p}_i$. However, each share of the codeword $\boldsymbol{c}_i^j = [\boldsymbol{m}_i^j | \boldsymbol{p}_i^j]$

is for itself not a valid codeword. The COMBINE-step is still necessary for error detection. To avoid this, the generation of the check bits need to be adjusted to

$$\boldsymbol{p}_i^j = \pi(\boldsymbol{m}_i^j),\ 1 \leq j \leq s. \tag{10.12}$$

Now each share of the codeword is valid and can be checked separately. In other words, each $\boldsymbol{p}_i^j$ can now be used to check its related $\boldsymbol{m}_i^j$ which makes the COMBINE-step unnecessary. Instead, if no error has occurred every $\boldsymbol{v}_i^j$ will be the null vector. To maintain this property, it is necessary that the predictors match exactly the main functions with additional encoding. Therefore, the aforementioned optimization technique regarding the decompositions of functions cannot be applied. Otherwise, the $\boldsymbol{p}_i^j$ would lose this characteristic and an additional COMBINE-step becomes necessary. It should be noted that this only works given that $n_p = n_m$. Otherwise, additional fresh randomness is required to achieve a uniform sharing of $\boldsymbol{p}_i$ making it impossible to check each share separately.

Overhead

The overhead of our scheme obviously depends on the chosen code and the underlying algorithm. Simple duplication, for example, is just an extreme case of our combined countermeasure in which P of the generator matrix is set to the identity matrix. However, if randomness is reused and $n_p \leq n$, the amount of fresh randomness is independent of the chosen code. In this case, our combined countermeasure uses the same amount of randomness as simple duplication. For other metrics it is not possible to give such a definite rule. Both area and performance can be worse or better than simple duplication, depending on how good the predictors can be optimized.

Combined Attacks

As mentioned in the introduction, there are combined attacks which can break AES implementations with certain combinations of countermeasures. Our proposed countermeasure can be also vulnerable to this kind of attacks depending on the underlying cipher and chosen code. However, most of these attacks focus on the error check and exploit that usually a combination of multiple shares is required. As described before our scheme can be instantiated without the necessity of a COMBINE-step which helps to prevent these attacks that rely on this as a point of attack. In this case, the leakage only contains information that an error has occurred but not more.

10.3.5 Security Analysis

We now discuss the security properties of our combined countermeasure under the previously defined attacker model. Here, we distinguish between resistance against SCA attacks and FI attacks. In the latter case, our combined countermeasure is generally compared with a simple duplication of the TI.

SCA Resistance

As mentioned before, the security of a TI is derived from its order. A first-order TI is provable secure against first-order attacks [NRR06]. Given that the adversary in our model can perform attacks up to order d, a TI of order d is accordingly required to protect our design. By following our proposed approach, the shared predictors are in compliance with the principle of a d-order TI. Therefore, they provide the same level of security as the d-order TI of the main circuit and our proposed combined countermeasure has the exact level of SCA-security as a plain d-order TI without FI countermeasures. Furthermore, this level is independent of the chosen code meaning that simple duplication does not provide better or worse SCA-protection than a more complex EDC.

FI Resistance

The level of security against FI attacks depends on the parameters of the chosen code. In particular, the code distance d is important for the detection of certain types of errors. In this context, we can model the simple duplication countermeasure as a linear $[2k, k, 2]$-code $\mathbf{D}$ with $d = 2$. This is a comparably low distance given that such a distance can be achieved by a (in most cases) much shorter parity $[k + 1, k, 2]$-code.

The efficiency of a fault countermeasure can be assessed by its fault coverage rate which measures the proportion of undetectable faults. To simplify the analysis, we first assume that the fault is injected into an intermediate state of the execution which is used for error detection. That is, $\boldsymbol{p}_{int}$ are valid check bits for $\boldsymbol{m}_{int}$. As defined before, a fault is modeled as an error vector $e \neq \mathbf{0}$ that is added to the state $\boldsymbol{c}_{int} = [\boldsymbol{m}_{int}|\boldsymbol{p}_{int}]$. For a fault to be undetectable, e needs to be a valid codeword of the deployed code $\mathbf{C}$. This is rooted in the characteristic of linear codes in which every valid codeword can be written as the sum of two valid codewords as

$$c_3 = c_1 + c_2 = m_1 \cdot G + m_2 \cdot G = (m_1 + m_2) \cdot G,$$

with $c_1, c_2, c_3 \in \mathbf{C}$. Therefore, if e is not a valid codeword of $\mathbf{C}$ the erroneous result would also not be a valid codeword. Note that the aforementioned addition property of linear codes still holds for shared codewords. Meaning that if a valid codeword is added to one of the shares, it would result in a new shared codeword. With this, we can formally define the fault coverage of a code $\mathbf{C}$ as

$$Coverage_{\mathbf{C}}[E \sim \mathcal{E}] = 1 - Pr[e \in \mathbf{C} \wedge e \neq \mathbf{0}], \tag{10.13}$$

where the error variable E follows an error distribution $\mathcal{E}$.

Usually the rank of the code k is not chosen to be equal to the size of the whole input of the algorithm for efficiency reasons. Therefore, the intermediate state of the execution consists of multiple valid codewords. To further simplify the analysis we first assume that the adversary only injects one fault in one share of one codeword of the intermediate state. With $|C| = 2^k$ and $|E| = 2^n$ we can derive $Pr[e \in \mathbf{C} \wedge e \neq \mathbf{0}] = (2^k - 1)/2^n$ and define the fault coverage of the code $\mathbf{C}$ as

$$Coverage_{\mathbf{C}}[E \sim \mathcal{E}_U] = 1 - \frac{2^k - 1}{2^n},$$

in the uniform fault model. Notably, the fault coverage in this model is independent of the code distance d. It means that it depends only on the rank k and the length n. Consequently, simple duplication provides the same fault coverage as any other code with the same k and n against this type of faults. For $\mathbf{D}$ the length is derived from the rank as $n = 2k$. The coverage can then be simplified to

$$Coverage_{\mathbf{D}}[E \sim \mathcal{E}_U] = 1 - \frac{1}{2^k} + \frac{1}{2^{2k}}.$$

As noted before, the uniform fault model is not a realistic assumption for all scenarios. Therefore, it is closer to reality to assume that the error distribution is biased to a certain degree [GMJK15]. For example, a clock glitch might cause similar errors in identical circuits which are close together (i.e., simple duplication). In this scenario, the fault coverage is severely reduced given that simple duplication cannot detect identical errors in both circuits. In the following, we assume that only a limited number of bits is affected by the fault. In the most extreme case, only one bit is affected which is related to laser fault injection[2]. We consider a biased distribution $\mathcal{E}_{B_b}$ with the corresponding subsets

$$E_1 = \{e \mid e \in E \wedge wt(e) \leq b\} \text{ with } Pr[e \in E_1] = 1, \tag{10.14}$$

$$E_2 = \{e \mid e \in E \wedge wt(e) > b\} \text{ with } Pr[e \in E_2] = 0. \tag{10.15}$$

We assume further that the error vectors in E_1 are all equally probable. Depending on the method of fault injection, certain values of b are easier to achieve than others. Following this definition, $\mathcal{E}_{B_n}$ is equivalent to $\mathcal{E}_U$. In this fault model, it is possible to give specific bounds in which a complete fault coverage is achieved by our proposed countermeasure. It is trivial to see that an $[n, k, d]$-code which can detect $u = d - 1$ errors still achieves a complete fault coverage in the model following $\mathcal{E}_{B_u}$. However, it depends on the specific code how the fault coverage evolves for higher values of $b > u$. For simple duplication it can be easily calculated as

$$Coverage_{\mathbf{D}}[E \sim \mathcal{E}_{B_b}] = 1 - \frac{\sum_{j=1}^{\lfloor \frac{b}{2} \rfloor} \binom{k}{j}}{\sum_{i=1}^{b} \binom{n}{i}}.$$

It is notable that a simple duplication scheme achieves full fault coverage only for $b = 1$.

Depending on the scenario, there are other possible biased distributions. For example, if the attacker is only able to inject faults in one part of the design (target algorithm or check bits) the full fault coverage is achieved for all codes with $d > 1$. Furthermore, the ability to inject symmetric errors in both parts strongly reduces the security of simple duplication. In the most extreme case, the adversary can pick bits to fault, e.g., by laser injection. In this case the error detecting capability is directly proportional to the attack complexity assuming that targeting more single bits by laser at different places increases the costs of the attack.

In reality, it might not be possible to only target one specific codeword, e.g., with round-based architectures. This affects the fault coverage since the error vector e needs to be valid codeword

---

[2] Note that bit flips which we assume in our attacker model might not be realistic for laser fault injection in certain scenarios [RSDT13]. However, we still use it in our model. The ability to set and reset bits instead of flipping enables trivial attacks in which the adversary tests each bit of the key to be zero or one. This attack cannot be directly prevented by our method without additional logic (e.g., allow only a certain number of faults). However, this is true for a majority of countermeasures and therefore not an issue unique to our methodology. The designer needs to include further countermeasures against this attack vector, e.g., splitting the key into multiple shares can increase the complexity of the attack.

for every element of the state. Therefore, the estimation of the coverage can be adapted to include the number of state elements n_s

$$Coverage_{\mathbf{C}}[E \sim \mathcal{E}] = 1 - (Pr[e \in \mathbf{C} \wedge e \neq \mathbf{0}])^{n_s} \, . \tag{10.16}$$

We assume an error check in which each share is not checked separately. Therefore, the number of shares does not play any role in this estimation since it is enough to check whether the sum of all shares is a valid codeword as

$$(c^1 + e^1) + (c^2 + e^2) + (c^3 + e^3) = c + e. \tag{10.17}$$

If each share is checked separately, the fault coverage needs to include the number of shares in the calculation similar to n_s.

Up to now, we only considered faults that are injected at one point in time into an encoded state which is checked for errors. Depending on the power of the adversary, this scenario may be realistic. However, there are other cases in which an attacker is more powerful and can inject more sophisticated types of errors.

One of these types are faults which are injected into a state between layers that is not directly checked. Instead, multiple operations are first performed on the erroneous state before it is checked. In this case, the fault coverage rate stays the same based on the fact that none of the operations change the validity of a codeword. In other words, if the error is detectable in one state, it should be also detectable in every following state.

Another important aspect for fault coverage is multiple faults at different points in time. Assume for example a linear transformation F of a codeword c in which an error e is injected before the transformation is applied. This results in

$$F(c + e) = F(c) + F(e) \tag{10.18}$$

meaning that the output of F is combined with a transformed error $F(e)$. Given the structure of the functions and predictors, $F(.)$ cannot make a valid codeword $F(e)$ if e is not a valid codeword. Therefore, the fault coverage is not impaired for faults at a single point in time. However, $F(.)$ can increase the Hamming weight of e making it easier for an attacker to inject an additional error after the transformation. In the most extreme case, an attacker injects only two errors e_1, e_2 with $wt(e_1) = wt(e_2) = 1$ and can create an undetectable fault as

$$F(c + e_1) + e_2 = F(c) + F(e_1) + e_2, \tag{10.19}$$

with $wt(F(e_1)) = d - 1$. This approach works similarly for non-linear layers. However, in this case the output of the function is not the sum of the two transformed values. This attack vector can be prevented by introducing more error checks in the design. If every encoded state before a transformation is checked, this attack can be thwarted since the injection of the first error $F(c+e_1)$ would be detected. Introducing more checks can obviously result in an increased area complexity.

As of now, all the errors are added to an encoded state between layers. However, depending on the scenario it might be also possible to inject faults inside the combinatorial logic between these states. Since the logic usually consists of a cascade of multiple gates modeling, the fault as an addition of an error vectors is not trivial. However, depending on the abilities of the attacker this type of fault can be powerful. For example, an attacker can target one gate which derives multiple output bits. In this case, we have the same scenario as in the previous example that the injected fault e has $wt(e) = 1$ but it cannot be detected when the check is performed on the output of the combinatorial circuit where such e leads to $e' \in \mathbf{C}$. To completely avoid this type of attack it is necessary to isolate the logic for all output lines from each other. This way a faulty gate can only affect one of the output bits which prevents the aforementioned attack.

As illustrated by the previous example, it is important to realistically estimate the power of potential FI attackers. Choosing a code with a large distance and implementing the previously proposed countermeasures might lead to a highly secure system. However, each of these aspects can negatively influence the size of the design. As for many other systems, the balance between area and the level of security is an important aspect in the design process.

10.4 Case Study: LED

Up to now, our combined countermeasure has been only discussed from the theoretic perspective without targeting a specific algorithm. To better illustrate the rationales and parameters of the design process, we implement a block cipher according to our methodology. For the sake of comprehensibility, a relatively straightforward example is picked to explain the design choices.

The most obvious target for this would be AES as it is the most widely deployed cipher. However, while the predictors for the linear layers of AES are comparably easy to implement, the TI of its non-linear layer poses still a challenge even without FI resistance [BGN$^+$14a]. In particular, it requires a significant amount of fresh randomness to achieve all the necessary TI properties. Another standardized cipher, for which an efficient TI exists, is PRESENT [BKL$^+$07]. Its 4-bit Sbox can be efficiently implemented in various ways [BNN$^+$12, SMG15a]. Contrary to AES, its permutation layer is very efficient in hardware, but its predictors are comparably inefficient. It consists of simple bit permutation over the whole 64-bit state and, therefore, does not require any gates. The predictor for this layer using the extended Hamming code can be found similar to Equation (10.29) by transforming the 64×64 permutation matrix. To this end the permutation matrix need to be twice multiplied with $\mathbf{P}$. In our scenario the two matrices P and $\mathbf{0}$ are defined as

$$P = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \text{ and } \mathbf{0} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{10.20}$$

While the original permutation matrix has only a single '1' per row, this is not the case for the predictors. Compared to the original permutation, the corresponding predictor requires several XOR gates to transform, split and merge the separate codewords. Therefore, it requires a non-negligible overhead to be implemented.

$$\mathbf{P} = \begin{pmatrix} P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & P \end{pmatrix}. \tag{10.21}$$

A better example to demonstrate our combined countermeasure is LED. It combined the best aspects of AES and PRESENT by incorporating the PRESENT Sbox and AES-like linear layers. Thus, an efficient TI and predictors can easily be achieved. In our case study, we present one way to implement LED with our methodology. Note that depending on the targeted attacker model, different choices are possible, e.g., higher-order TI or another code with a large distance. The SCA security of the final design is practically evaluated using an FPGA prototype, while the FI resistance is examined using the previously introduced attacker models.

10.4.1 Cipher Description

LED is a lightweight block cipher introduced in 2011 [GPPR11]. It has a 64-bit state and can be instantiated with different key sizes (primarily 64 or 128 bits). The basic structure of the cipher as depicted in Figure 10.5 consists of addition of the round keys (addRoundKey) and so-called steps (step). In each step, four rounds of encryption are applied to the state. One round is made up of four layers AddConstants, SubCells, ShiftRows and MixColumnsSerial. During AddConstants constants which are derived from an Linear Feedback Shift Register (LFSR) are added to half of the state. The following three layers are similar to the layers of AES [Pub01] and consist of a nibble-wise substitution and row/column-wise affine transformations. For 128-bit key (resp. 64-bit keys) LED-128 (resp. LED-64) performs 12 steps in total (resp. 8 steps) with key additions between them.

One important characteristic of LED is its very simple key schedule. Instead of using different round keys derived by a schedule function applied on a main key, the cipher directly uses 64 bits from the user-defined key for each round. This means that for the 64-bit version all round keys are the same, while in the 128-bit instantiation the key halves are used alternately.

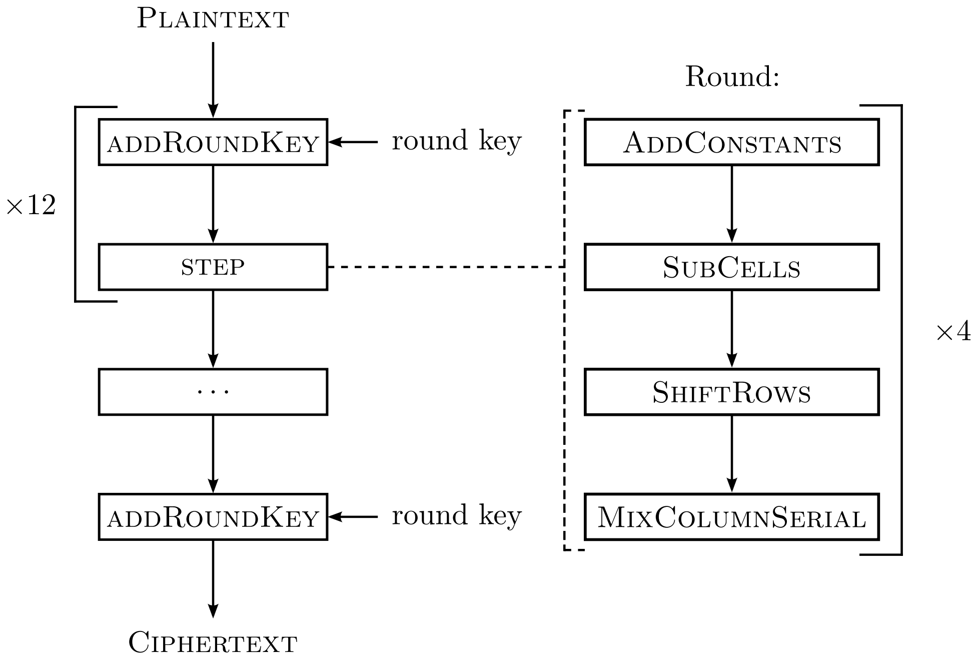


Figure 10.5: The basic structure of LED-128.

10.4.2 Design and Implementation

We implement a design that is secure against first-order attacks. We decompose the Sbox that allows us to implement TI using three shares. In the following, we explain the selection of the code and the predictors for each layer of LED in detail.

Code Selection.

Given that LED is a nibble-oriented cipher in which all operations work on either one or multiple nibbles of the state, we consider only codes with a rank of $k = 4$. This way, expensive merge or split of codewords can be minimized. Furthermore, we decided to set the length of the code to $n = 8 = 2 \cdot k$ to avoid additional fresh randomness. It would be beneficial to select a code over $GF(2^4)$, since most of the LED operations are in this field[3]. However, none of the 16 possible $[8, 4]$-codes has a distance larger than $d = 3$. Therefore, to achieve a higher level of protection against FI attacks, we choose a different code outside of $GF(2^4)$ but with a better error detection property.

---

[3] In this case, P is chosen in such a way that $\boldsymbol{p} = \pi(\boldsymbol{m}) = \boldsymbol{m} \cdot x$ with $x \in GF(2^4)$.

The extended Hamming code is a basic extension of the $[7, 4, 3]$-Hamming code. By adding an extra parity bit the code is transformed to a $[8, 4, 4]$-code, i.e., with $d = 4$. In our implementation we use the following generator and parity check matrices:

$$G = \left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right), \qquad H = \left(\begin{array}{cccc|cccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right). \tag{10.22}$$

Due to its simplicity, the code enables the use of efficient predictors while still achieving a high error detection capability with respect to its length.

Linear Layers.

As described before, LED consists of four different linear layers. We discuss the application of the extended Hamming code to each layer without specifically considering TI, since every linear layer and corresponding predictor can be applied to each share separately as explained in Section 10.3.4. Note that the key and constants are not shared, following the same design strategy as in [BGN⁺14a, MPL⁺11, PMK⁺11, SMG15a]. Therefore, in the two layers (AddRoundKey, AddConstants) where a value is added to the state, it is applied only to one share (of three).

AddRoundKey. Since this layer only consists of a basic addition in $GF(2^4)$ of the round key to the state of the cipher, its predictor can be implemented very efficiently. It can be optimized to

$$\begin{aligned} \boldsymbol{p}_{int_2} &= \pi\left(\pi^{-1}\left(\boldsymbol{p}_{int_1}\right) \oplus key\right) \\ &= \boldsymbol{p}_{int_1} \oplus \pi\left(key\right), \end{aligned} \tag{10.23}$$

where $\boldsymbol{p}_{int_1}$ (resp. $\boldsymbol{p}_{int_2}$) denotes the input (resp. output) check bits to AddRoundKey, and key a round key. Furthermore, LED does not include a key schedule. Thus, by computing $\pi\left(key\right)$ (of both key halves for LED-128) once at the start of the cipher, the predictor for the key addition can be easily realized without additional overhead.

AddConstants. Two types of round constants are added to the state. One is derived from the key length and does not change over the course of the cipher. The bit size of the key length is stored in eight bits ($ks_7 ks_6 ks_5 ks_4\ ks_3 ks_2 ks_1 ks_0$). The lower and upper four bits of the bitstring are each considered as one encoded element. Since the key size does not change during the execution, this type of constant does not need to be updated. For LED-128 the specific bits are

$$\begin{aligned} (ks_7 ks_6 ks_5 ks_4\ ks_3 ks_2 ks_1 ks_0) &= (1000\ 0000), \\ (ksp_7 ksp_6 ksp_5 ksp_4\ ksp_3 ksp_2 ksp_1 ksp_0) &= (1110\ 0000) \end{aligned} \tag{10.24}$$

where ksp_i denotes the corresponding check bits for this constant.

The other constant consists of six bits $(rc_5rc_4\ rc_3rc_2rc_1rc_0)$ which are updated for every round by an LFSR. The update function can be represented by a matrix multiplication in $GF(2)$ as

$$\begin{pmatrix} rc'_0 \\ rc'_1 \\ rc'_2 \\ rc'_3 \\ rc'_4 \\ rc'_5 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}}_{U} \cdot \begin{pmatrix} rc_0 \\ rc_1 \\ rc_2 \\ rc_3 \\ rc_4 \\ rc_5 \end{pmatrix} + \underbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{c}, \tag{10.25}$$

where U denotes the update matrix. The related check bits defined as

$$(rcp_3rcp_2rcp_1rcp_0) = \pi(rc_3rc_2rc_1rc_0) \tag{10.26}$$

$$(rcp_7rcp_6rcp_5rcp_4) = \pi(00|rc_5rc_4) \tag{10.27}$$

need to be updated accordingly. To this end, the update matrix is first enlarged to incorporate the two padded zeros to

$$U_L = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{10.28}$$

The update matrix for the check bits $(U_{L_{check}})$ can be derived from by $\pi\left(U_L\left(\pi^{-1}\left(\cdot\right)\right)\right)$. Therefore, we can write (note that P is self-inverse):

$$U_{L_{check}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \cdot U_L \cdot \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}. \tag{10.29}$$

The same procedure is applied to the constant factor of the update function (denoted as c in Eq. (10.25)). Overall, the check bits of the round constant can be updated as

$$\begin{pmatrix} rcp'_0 \\ rcp'_1 \\ rcp'_2 \\ rcp'_3 \\ rcp'_4 \\ rcp'_5 \\ rcp'_6 \\ rcp'_7 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}}_{U_{L_{check}}} \cdot \begin{pmatrix} rcp_0 \\ rcp_1 \\ rcp_2 \\ rcp_3 \\ rcp_4 \\ rcp_5 \\ rcp_6 \\ rcp_7 \end{pmatrix} + \underbrace{\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{cp} . \tag{10.30}$$

It is obvious that the update of the check bits requires additional resources. Still, this overhead is negligible since the round constant update is only a small part of the cipher and not split up into multiple shares.

ShiftRows. This layer manipulates the state in a nibble-wise fashion. Since the codewords are not modified in any way, it is sufficient to apply the same permutation on the check bits.

MixColumnSerial. Four nibbles of the state are combined using a matrix A four consecutive times. The matrix multiplication is performed in $GF(2^4)$. Since addition is linear over $GF(2)$, we do not need to change the values of A for the check bits. Only the field multiplications with 2 and 4 need to be adapted to the predictor. The two multiplications with the reduction polynomial $X^4 + X + 1$ can be represented as a matrix multiplications in GF(2) as

$$2 \cdot \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \end{pmatrix}, \qquad 4 \cdot \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \end{pmatrix}. \tag{10.31}$$

For the check bits, these matrices need to be adapted similar to Equation (10.29) but with 4×4 matrices. The resulting matrices for the check bits are

$$\pi\left(2 \cdot \pi^{-1} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}\right) = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}, \quad \pi\left(4 \cdot \pi^{-1} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}\right) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}. \tag{10.32}$$

This layer is also slightly more costly for the check bits. However, the overhead is not as significant as for PRESENT.

Non-Linear Layer.

Similar to [PMK$^+$11], we decomposed the Sbox into two steps to reduce the number of required shares to three. The functions for the state and check bits are optimized independently of each other. As mentioned before, this procedure results in a more efficient implementation in terms of area with the penalty of not being able to check the correctness of each share individually. To find an area-efficient representation, we applied the same idea as in [BNN+12]. In particular, different affine transformations with different combinations of quadratic bijective classes (as defined in [BNN+15]) are tested and compared by their number of XOR and AND operations [PMK+11]. For the non-encoded TI we tested combinations of the form

$$S = A_3 \circ T_2 \circ A_2 \circ T_1 \circ A_1, \tag{10.33}$$

where A_1, A_2, A_3 are affine transformations and T_1, T_2 are quadratic bijections. We tested all possible valid combinations of Table 1 from [SMG15a] and decomposed the Sbox as $S(m) = F\left(G\left(x\right)\right), \forall m$ with

$$F = A_3 \circ T_2, \qquad G = A_2 \circ T_1 \circ A_1.$$

For the check bits Equation (10.33) is slightly adjusted to

$$S_p = \pi \circ S \circ \pi^{-1} = \pi \circ A_3 \circ T_2 \circ \pi^{-1} \circ \pi \circ A_2 \circ T_1 \circ A_1 \circ \pi^{-1}, \tag{10.34}$$

and the Sbox for the check bits is split as $S_p(p) = Q\left(R\left(p\right)\right), \forall p$ with

$$Q = \pi \circ A_3 \circ T_2 \circ \pi^{-1}, \qquad R = \pi \circ A_2 \circ T_1 \circ A_1 \circ \pi^{-1}.$$

We found the most efficient decomposition for the classical TI using the quadratic class $\mathcal{Q}_{12}$ for both T_1, T_2 (see [BNN+12]). For the check bits the most efficient decomposition was obtained by the quadratic classes $\mathcal{Q}_{294}$ and $\mathcal{Q}_{299}$ for T_1 and T_2, respectively.

As a side note, since $R \neq \pi \circ G \circ \pi^{-1}$ (and likewise for Q and F), the error-checking procedure cannot be performed in-between the Sbox computation. Below we list the algebraic normal form (ANF) of the derived (and applied) functions (a and e as least significant bits).

$$\begin{aligned}
G(d,c,b,a) = (h,g,f,e):\quad & e = a + c + d + cb & f &= a && (10.35)\\
& g = 1 + a + d + b + cb & h &= 1 + a + bc + bd + cd &&\\
F(d,c,b,a) = (h,g,f,e):\quad & e = a & f &= c + d + bd && (10.36)\\
& g = 1 + a + b + c + cd & h &= c + bd &&\\
R(d,c,b,a) = (h,g,f,e):\quad & e = a + b + db + dc & f &= b + c && (10.37)\\
& g = c + ba + ca & h &= d + b + cb &&\\
Q(d,c,b,a) = (h,g,f,e):\quad & e = 1 + a + b + c + db + dc & f &= 1 + a + b && (10.38)\\
& g = a + d + db + dc & h &= c + ab + ac + ad + bc + bd &&
\end{aligned}$$

The uniform shared representations of the component functions (G_1, G_2, G_3), (F_1, F_2, F_3), (R_1, R_2, R_3), (Q_1, Q_2, Q_3) can be derived by direct sharing [BNN+15]. All required formulas are given in Appendix 15.1.

Basic Structure.

We implemented the LED encryption with our countermeasure following a round-based architecture. The basic structure of our design is depicted in Figure 10.6 with the predictors in the

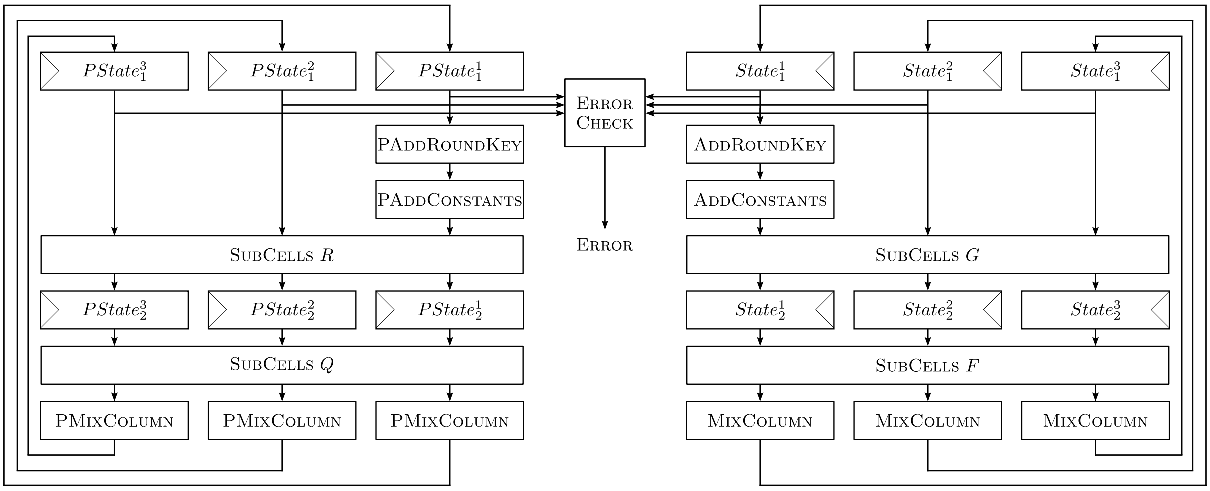


Figure 10.6: The basic structure of our proposed LED design. Multiplexers for the plaintext and ADDROUNDKEY are omitted.

left half. As stated above, the Sbox and its corresponding function on check bits do not follow the same decomposition. Therefore, we perform the error check only at the first registered state $State_1^{i\in\{1,2,3\}}$. The ERROR CHECK module has been implemented following the concept of CHECK-AND-COMBINE, illustrated in Section 10.3.4. Both ADDROUNDKEY and ADDCONSTANTS are only applied to the first share since the key and the constants are not shared. An additional register stage is necessary inside SUBCELLS (between G and F as well as between R and Q) to avoid the propagation of glitches. The initial randomness is shared between both parts of the circuit and none of the layers requires additional fresh randomness to achieve uniformity. It should be noted that except for the initial loading (right half with shared plaintext, and left half with shared corresponding check bits) the two halves of the design do not interact with each other, and each one operates independently. At every clock cycle, the ERROR CHECK module examines the consistency of the state and its corresponding check bits.

The proposed design can be easily extended to provide security against higher-order attacks by increasing the number of shares. As the linear functions are applied on each share separately, their basic structure does not change, while non-linear functions require further adjustment. A second-order TI of the PRESENT Sbox is given in [MW15]. However, mask refreshing might be necessary to ensure resistance against multivariate higher-order attacks as indicated in [RBN$^+$15]. Note, however, that for higher-order TI the error check needs to be also adjusted accordingly to comply with the TI properties. In other words, extra registers should be integrated into COMBINE step of CHECK-AND-COMBINE module (see Figure 10.4) and the COMBINE should be performed in several clock cycles to ensure that the desired higher-order resistance is not violated.

10.4.3 Area Comparison

We synthesized our implementations with the Synopsys Design Compiler using the UMC-L18G212T3 [Vir04] ASIC standard cell library (UMC 0.18μm). The results are presented in Table 10.1.

Table 10.1: Size of our design for an ASIC platform.

Module	**Area [GE]**			
	Original	Predictors	Error Detection	Control
AddRoundKey	171	171	-	-
AddConstants	32	44	-	-
SubCells 1	1750	1584	-	-
SubCells 2	1051	2795	-	-
ShiftRows	0	0	-	-
MixColumnSerial	1532	2048	-	-
Total	7891	10028	2023	270
LED-ParTI	**20212**			

As expected, the state registers constitute a significant portion of each circuit part (in the following referred to as *Original* and *Predictors*). Furthermore, the decomposed Sbox is in both cases the largest layer of the design. Since we make use of CHECK-AND-COMBINE, the error detection circuitry is relatively large due to the required additional registers of the COMBINE step. Overall, the predictors require around 27% more area than the original TI. With the same error detection module, our design with the extended Hamming code is around 12% bigger than simple duplication.

The synthesized circuit can operate at the maximum frequency of 148 MHz and requires 96 clock cycles for one encryption. The design forms a pipeline, where two plaintexts can be consecutively fed. This results in a maximum throughput of 197.3 Mbit/s. In comparison, the unprotected round-based implementation requires 46 clock cycles for one encryption and can operate at a maximum frequency of 131 MHz. This results in a throughput of 174.7 Mbit/s since the design does not allow a pipeline.

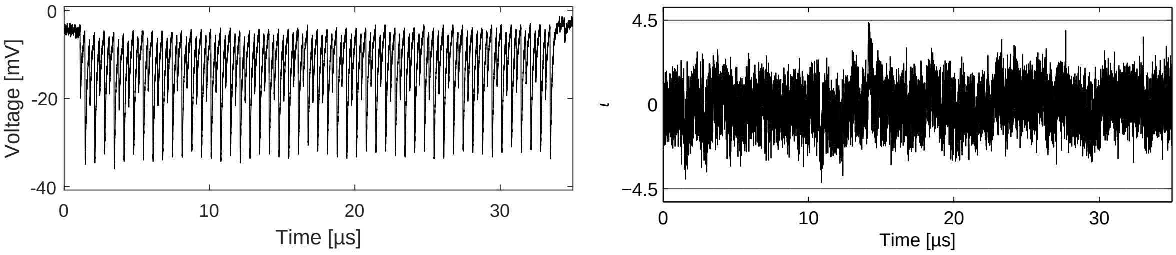
10.4.4 Resistance against SCA

Given that all functions are compliant to the principles of TI, our design is provably secure against first-order attacks. Nevertheless, we also evaluated the security of our design experimentally using an FPGA and ported our design to the FPGA-based side-channel evaluation platform SAKURA-G [Sak] populated with a Xilinx Spartan-6 FPGA. The power traces obtained for our the design have been collected by means of a digital oscilloscope at sampling rate of 500 MS/s while the design was operating at a frequency of 3 MHz.

As an evaluation metric, we used the non-specific t-test as described in Chapter 4. Figure 10.7 depicts the results for univariate tests at first, second and third orders using 100 million measurements. The diagram show that our design is indeed first-order secure while – as expected – leakages for higher orders can be observed.

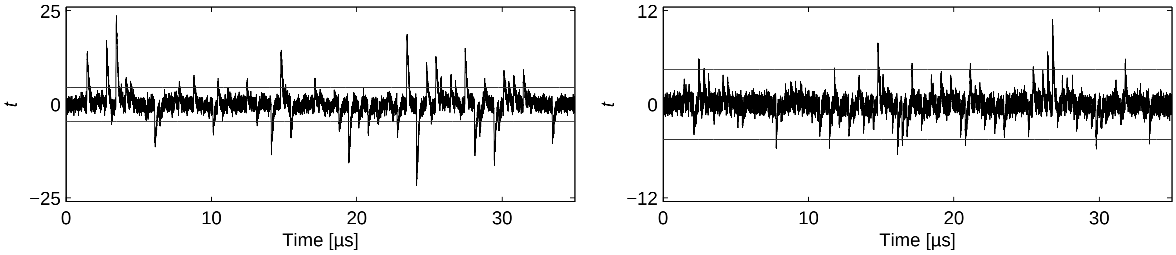
10.4.5 Resistance against FI

We further examined the fault coverage of our scheme considering the previously introduced attacker model. Given that the extended Hamming $[8, 4, 4]$-code has a distance of $d = 4$, it



(a) trace

(b) first order



(c) second order

(d) third order

Figure 10.7: A sample trace and the result of non-specific t-tests at orders one to three.

can detect errors up to $wt(e) \leq u = 3$. The coverage of this code is compared to the coverage that can be achieved with a simple $[8,4,2]$-duplication code with $u = 1$. We compute the fault coverage of both codes for the uniform distribution as well as for biased distributions $\mathcal{E}_{B_1}$ to $\mathcal{E}_{B_8}$. We consider both the best case (BC) and worst case (WC) for an attacker. In the best case, the attacker is able to inject a fault into one share of a single codeword. In the worst case, he can inject faults into all shares of all codewords simultaneously. Given that the TI of LED operates on a 16-element state, the fault coverage is significantly increased in this case. Since we do not check each share separately, the number of shares does not influence the fault coverage rate of the worst case.

Table 10.2 represents the fault coverage rates for the examined cases for both codes. We already discussed how to compute the fault coverage for a duplication code in Section 10.3.5. In order to derive the corresponding fault coverage for the $[8,4,4]$-code, we look at the distribution of the Hamming weight of the codewords. Since $k = 4$, there exist 16 different codewords. 14 of them have Hamming weight $wt(c) = 4$, while there are two codewords with $wt(c) = 0$ and $wt(c) = 8$ respectively. Therefore, only some error vectors with Hamming weight of 4 or 8 (excluding the zero error vector) have the possibility to be undetectable by our scheme[4].

This observation is confirmed by the results in Table 10.2. The $[8,4,4]$-code provides full fault coverage in the biased model up to $\mathcal{E}_{B_3}$. Given that most of the valid codewords have a Hamming weight of 4, and $d = 4$, the biased distribution $\mathcal{E}_{B_4}$ leads to the lowest fault coverage. In short, $\mathcal{E}_{B_4}$ and $\mathcal{E}_{B_5}$ are the only distributions with which the simple duplication scheme is better than the extended Hamming code. For all other cases, the extended Hamming code outperforms (or is equal to) the simple duplication scheme. As expected, the worst case leads

---

[4] Since $d = 4$, errors which flip 4 or 8 bits can turn a valid codeword into another valid codeword, and are hence undetectable.

Table 10.2: Fault coverage for different distributions and codes.

	$[\mathbf{8},\mathbf{4},\mathbf{4}]$		$[\mathbf{8},\mathbf{4},\mathbf{2}]$	
	BC	WC	BC	WC
$\mathcal{E}_U$	0.94	$1-2^{-65}$	0.94	$1-2^{-65}$
$\mathcal{E}_1$	1.00	1.00	1.00	1.00
$\mathcal{E}_2$	1.00	1.00	0.89	$1-2^{-51}$
$\mathcal{E}_3$	1.00	1.00	0.95	$1-2^{-72}$
$\mathcal{E}_4$	0.91	$1-2^{-56}$	0.93	$1-2^{-64}$
$\mathcal{E}_5$	0.93	$1-2^{-63}$	0.95	$1-2^{-71}$
$\mathcal{E}_6$	0.94	$1-2^{-66}$	0.94	$1-2^{-66}$
$\mathcal{E}_7$	0.94	$1-2^{-66}$	0.94	$1-2^{-66}$

to very high fault coverage given that the probability to inject an error which results in a valid codeword for every element of the state is very low.

10.5 Conclusions and Future Work

We presented an advanced hardware countermeasure which offers resistance both against SCA and FI attacks. In short, we proposed a construction to combine error detecting codes with the concept of threshold implementations. We have identified and discussed generic strategies to that additions for information redundancy do not contradict to the assumptions and requirements of the underlying masking scheme.

From a general point of view, our combined countermeasure can be applied to arbitrary ciphers and supports different level of protections, i.e., first- or higher-order SCA resistance as well as various fault coverage settings. As an example, we have illustrated how to apply our methodology on the LED block cipher with the aim of maintaining first-order SCA protection while integrating an extended Hamming code to detect faults. Supported by our experimental validation, we have demonstrated how to realize an efficient design that satisfies the requirement to provide protection against SCA and FI.

These results lay the foundation for future research on various interesting topics. Currently, the fault resistance of the schemes is evaluated based on the theoretical properties of the used error-detecting code. Another approach would be to profile the target device using practical measurements and estimate the actual error distributions. From these results, the error-detecting code with the higher fault coverage could be easily derived. Furthermore, it would be interesting to apply the proposed methodology to the widely-used AES. However, the TI of the corresponding 8-bit Sbox still requires much additional randomness which would make a protected design with our methodology challenging. The same problem applies to the extension of our case study to (multivariate) higher-order security.

Part IV

Conclusion

Chapter 11

Conclusion and Future Work

In this chapter, we summarize our results regarding our advances in both side-channel evaluation techniques and the design of hardware-based countermeasures against physical attacks. Additionally, we discuss interesting future research opportunities.

Contents of this Chapter

11.1 Conclusion **155**
11.2 Future Work **157**

11.1 Conclusion

The first half of this thesis was dedicated to the evaluation of countermeasures. Validation of the practical security is indispensable for a final product and can help to fairly compare protection schemes in the design phase with the goal of selecting the hardening strategy with the best cost efficiency. We contributed to further the state-of-the-art of three commonly-used leakage evaluation methodologies. The first chapter of this part [SM15a, SM16] extended the theoretical foundation of the popular Test Vector Leakage Assessment (TVLA) methodology [CDG^{+}13, GJJR11] based on Welch's t-test. Our contributions enabled the methodology to cover more classes of sophisticated countermeasures (multivariate and higher-order evaluations), proposed the use of incremental formulas to increase the computation speed with improved accuracy, and presented guidelines to design a correct and highly-efficient measurement environment which is supported by two case studies. This resulted in a very efficient first test to check the validity of the underlying assumptions of a countermeasure, but lacks more advanced intuitions on the distribution of the leakage. For a more thorough assessment, information-theoretic evaluations are often conducted as a subsequent measure [SMY09]. Our contribution [SMSG16a] increased the resolution of these evaluations helping the evaluator to more efficiently assess the leakage of the device and analyze the integrated countermeasure. As a final step, attack-based evaluation are often used to derive a practical security level, e.g., as part of a formal certification process. However, the large number of different types of possible attack vectors make an exhaustive evaluation infeasible. In this thesis, we proposed to apply incremental computation to significantly reduce the computationally complexity and increase accuracy of correlation-based attacks [SMG16c]. Since Pearson's correlation coefficient accounts for a large share of all attacks, this improved the efficiency of attack-based evaluations significantly and enabled the rapid testing of protected hardware designs.

In the second part of this thesis, we furthered the state-of-the-art in the design of hardware-based countermeasures against physical attacks. Firstly, we examined the problem of provably protecting integer addition in hardware against side-channel analysis [SMG15b]. While this seems like a fundamental operation and many software-oriented solutions [Gou01, KRJ14] exist, there were no hardware studies offering provable security supported by practical verification. We presented the first secure addition circuits working with Boolean-masked inputs and evaluated their security. This enabled the secure and efficient hardware design of primitives that require integer addition combined with Boolean operations, e.g., ChaCha [Ber08a] and Skein [FLS^{+}10]. Secondly, we explored an alternative design paradigm [BGG^{+}16a] for threshold implementations. Instead of applying TI to a given algorithm which can result in non-trivial transformations due to the high non-linearity of certain operations, we proposed to invert the development process by creating good cryptographic permutations from smaller functions which can be efficiently protected by TI. Our constructions were equally strong as commonly-used permutations (excluding AES), but their resource consumption, throughput, and latency was considerably better than the state-of-the-art. These results are especially useful for cipher designers as they promote the design of high-security block ciphers with intrinsic protection against physical attacks. Thirdly, the most important contribution of this thesis consisted of a combined countermeasure against both types of physical attacks [SMG16b]. In practice, the integration of countermeasures against only side-channel analysis (as described in the first two parts of contribution) is not sufficient to provide reasonable security against a physical adversary, as sensible information can be still be extracted via fault injection. Therefore, without the integration of dedicated countermeasures against both active and passive physical attacks, embedded systems should not be utilized to process sensitive information. While many countermeasures against each type have been proposed over the years, a majority of these publications focuses only one of the two types and, accordingly, the impact of the proposed countermeasures is not evaluated in the complementary context. This results in inefficient combinations of independent countermeasures and in the worst-case can negatively affect the security due to one countermeasure canceling another. First approaches were made to design a combined countermeasure based on coding theory by Bringer *et al.* in [BCC^{+}14]. However, their solutions are optimized for software implementations and cannot be easily transferred to a dedicated hardware circuits without sacrificing security or efficiency. In this thesis, we presented a new methodology to design hardware-based implementations protected against both types of physical attacks by combining the concept of threshold implementations with the capability of error-detecting codes. Our methodology guides the designer through the whole development process and is easily scalable to higher security levels. We applied our approach in a case study targeting the LED cipher [GPPR11] and practically verified the side-channel security of our design. The final circuit traded a small increase in area for significantly better protection against realistic fault models compared to more simplistic approaches. This represents the first comprehensive hardware study and our work can be seen as a sound basis for further research in the previously neglected field of hardware-based combined countermeasures.

Concluding, physical attacks are an imminent threat to any system which provides physical access to an adversary. Therefore, a security-aware design process includes the integration and validation of countermeasure against such an adversary to protect sensitive information inside the hardware circuit from unauthorized extraction. The proposed evaluation and design methodologies solve some open issues related to hardware-based countermeasures, since dedicated hardware circuits are an essential part of future embedded systems. Both directions benefit a security-aware design process by increasing its security (i.e., adjusting the adversary model to be closer to reality and a more thorough leakage assessment) and efficiency (i.e., more efficient design methodologies and a faster leakage assessment for comparison). Some results already served as a basis for future research, e.g., one of our protected addition circuits is used for the side-channel protected processor SPARX published by Bache *et al.* [BSMG17] and our leakage assessment methodology is widely used to quickly verify the security of new countermeasures. Furthermore, our combined scheme also spawned new research (e.g., [SES17, RMB$^+$17]), which further underlines the impact of our results. Nevertheless, there are still many open issues related to physical attacks and countermeasures that need to be examined. Some related to our results are discussed in the next subsection.

11.2 Future Work

After describing specific possible extension for our contributions in the corresponding chapters, we discuss more general ideas for future work in the following.

11.2.1 Physical Adversary Model

In many publications, the probing model [ISW03] is used to prove the security of masked algorithms. While this provides a simple way to evaluate new algorithms, the model is based on idealistic assumptions and does not take all physical peculiarities into account, e.g., glitches or transitional leakage. This often leads to a gap between the security of an algorithm in the probing model and the practical security of its implementation [BGG$^+$14]. Recently, first advances have been made to extend the probing model and close this gap [FGP$^+$17]. While this extended probing model includes many physical effects, it does not yet consider active adversaries. Future work would see the development of an encompassing physical model in which the security against a “realistic” combined (active and passive) adversary can be efficiently shown. In this context, a similar notion to t-SNI [BBD$^+$16] is interesting as it would enable the composability of smaller parts into a big circuit.

11.2.2 Efficient and Secure Randomness Generation

Most side-channel countermeasures require some form of randomness to provide the desired level of security. However, the problem of generating enough randomness securely is often overlooked in the related literature. Without dedicated countermeasures, the random number generator is an attractive target for a potential attacker as it allows to easily disable the countermeasures of the target circuit. Therefore, this is an interesting field of future research which can significantly improve the security and efficiency of the whole system. One starting point is related to the aforementioned physical adversary model. It can be used to formally show the security of the newly designed randomness generator circuit. The same notion also transfers to the protection of control logic (especially the comparison) of fault countermeasures. Without protecting every essential component of the target circuit, the implemented countermeasures will not have any effect on the security.

11.2.3 Masking in Hardware with Less Shares

Recent publications [CRB$^+$16, CFE16, GMK16, GM17] have proposed hardware masking schemes with only $d+1$ shares instead of the $td+1$ shares of threshold implementation. This reduction comes with significant improvements in terms of area and randomness especially for higher-order implementations. However, the authors of [CFE16] showed that the practical security of these schemes is also reduced by the reduced number of shares. Future research should include a more thorough evaluation of the approach with a fair comparison of the three masking schemes and more evaluations of the practical security compared to traditional threshold implementations. In this context, it also would be interesting to see if our designs could be improved by replacing the threshold implementation with some of the new hardware masking schemes.

11.2.4 Physically Secure Cipher

There have been previous approaches to design block ciphers with a particular focus on physical protection, e.g., FIDES [BBK$^+$13] which can be easily implemented with TI. However, there were no designs which try to include some kind of fault detection mechanism to thwart fault attacks. Therefore, designing a cipher which intrinsically provides resistance against passive and active physical attacks is an interesting research direction. We can build on our results from Chapter 9 and extend the search to include a fault resistance countermeasure similar to Chapter 10. These new Sboxes then serve as a foundation of a physically secure cipher.

Part V

Appendix

Chapter 12

Specific Formulas

12.1 Univariate Two-Pair Iterative

Below we consider $\Delta_t = \mu_{t,\mathcal{Q}_2} - \mu_{t,\mathcal{Q}_1}$ and $\Delta_l = \mu_{l,\mathcal{Q}_2} - \mu_{l,\mathcal{Q}_1}$ with $|\mathcal{Q}_1| = n_1$ and $|\mathcal{Q}_2| = n_2$. The extended set is defined as $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$ with the cardinality of n. The computations related to the sample points should be repeated at each sample point separately.

12.1.1 Central Sums

$$CS_{2,\mathcal{L}} = CS_{2,\mathcal{L}_1} + CS_{2,\mathcal{L}_2} + n_1\, n_2 \Big(\frac{\Delta_l}{n}\Big)^2 \Big(n_1 + n_2\Big) \tag{12.1}$$

$$CS_{2,\mathcal{Q}} = CS_{2,\mathcal{Q}_1} + CS_{2,\mathcal{Q}_2} + n_1\, n_2 \Big(\frac{\Delta}{n}\Big)^2 \Big(n_1 + n_2\Big) \tag{12.2}$$

$$\begin{aligned} CS_{3,\mathcal{Q}} = {} & CS_{3,\mathcal{Q}_1} + CS_{3,\mathcal{Q}_2} + \frac{3\,\Delta}{n}\Big(- n_2\, CS_{2,\mathcal{Q}_1} + n_1\, CS_{2,\mathcal{Q}_2}\Big) \\ & + n_1\, n_2 \Big(\frac{\Delta}{n}\Big)^3 \Big({n_1}^2 - {n_2}^2\Big) \end{aligned} \tag{12.3}$$

$$\begin{aligned} CS_{4,\mathcal{Q}} = {} & CS_{4,\mathcal{Q}_1} + CS_{4,\mathcal{Q}_2} + \frac{4\,\Delta}{n}\Big(- n_2\, CS_{3,\mathcal{Q}_1} + n_1\, CS_{3,\mathcal{Q}_2}\Big) \\ & + 6\Big(\frac{\Delta}{n}\Big)^2 \Big({n_2}^2\, CS_{2,\mathcal{Q}_1} + {n_1}^2\, CS_{2,\mathcal{Q}_2}\Big) + n_1\, n_2 \Big(\frac{\Delta}{n}\Big)^4 \Big({n_1}^3 + {n_2}^3\Big) \end{aligned} \tag{12.4}$$

$$\begin{aligned} CS_{5,\mathcal{Q}} = {} & CS_{5,\mathcal{Q}_1} + CS_{5,\mathcal{Q}_2} + \frac{5\,\Delta}{n}\Big(- n_2\, CS_{4,\mathcal{Q}_1} + n_1\, CS_{4,\mathcal{Q}_2}\Big) \\ & + 10\Big(\frac{\Delta}{n}\Big)^2 \Big({n_2}^2\, CS_{3,\mathcal{Q}_1} + {n_1}^2\, CS_{3,\mathcal{Q}_2}\Big) + 10\Big(\frac{\Delta}{n}\Big)^3 \Big(- {n_2}^3\, CS_{2,\mathcal{Q}_1} + {n_1}^3\, CS_{2,\mathcal{Q}_2}\Big) \\ & + n_1\, n_2 \Big(\frac{\Delta}{n}\Big)^5 \Big({n_1}^4 - {n_2}^4\Big) \end{aligned} \tag{12.5}$$

$$\begin{aligned} CS_{6,\mathcal{Q}} = {} & CS_{6,\mathcal{Q}_1} + CS_{6,\mathcal{Q}_2} + \frac{6\,\Delta}{n}\Big(- n_2\, CS_{5,\mathcal{Q}_1} + n_1\, CS_{5,\mathcal{Q}_2}\Big) \\ & + 15\Big(\frac{\Delta}{n}\Big)^2 \Big({n_2}^2\, CS_{4,\mathcal{Q}_1} + {n_1}^2\, CS_{4,\mathcal{Q}_2}\Big) + 20\Big(\frac{\Delta}{n}\Big)^3 \Big(- {n_2}^3\, CS_{3,\mathcal{Q}_1} + {n_1}^3\, CS_{3,\mathcal{Q}_2}\Big) \\ & + 15\Big(\frac{\Delta}{n}\Big)^4 \Big({n_2}^4\, CS_{2,\mathcal{Q}_1} + {n_1}^4\, CS_{2,\mathcal{Q}_2}\Big) + n_1\, n_2 \Big(\frac{\Delta}{n}\Big)^6 \Big({n_1}^5 + {n_2}^5\Big) \end{aligned} \tag{12.6}$$

$$\begin{aligned}CS_{7,\mathcal{Q}} = {} & CS_{7,\mathcal{Q}_1} + CS_{7,\mathcal{Q}_2} + \frac{7\,\Delta}{n}\Big(- n_2\, CS_{6,\mathcal{Q}_1} + n_1\, CS_{6,\mathcal{Q}_2}\Big) \\ & + 21\Big(\frac{\Delta}{n}\Big)^2\Big({n_2}^2\, CS_{5,\mathcal{Q}_1} + {n_1}^2\, CS_{5,\mathcal{Q}_2}\Big) + 35\Big(\frac{\Delta}{n}\Big)^3\Big(- {n_2}^3\, CS_{4,\mathcal{Q}_1} + {n_1}^3\, CS_{4,\mathcal{Q}_2}\Big) \\ & + 35\Big(\frac{\Delta}{n}\Big)^4\Big({n_2}^4\, CS_{3,\mathcal{Q}_1} + {n_1}^4\, CS_{3,\mathcal{Q}_2}\Big) + 21\Big(\frac{\Delta}{n}\Big)^5\Big(- {n_2}^5\, CS_{2,\mathcal{Q}_1} + {n_1}^5\, CS_{2,\mathcal{Q}_2}\Big) \\ & + n_1\, n_2\Big(\frac{\Delta}{n}\Big)^7\Big({n_1}^6 - {n_2}^6\Big)\end{aligned} \tag{12.7}$$

$$\begin{aligned}CS_{8,\mathcal{Q}} = {} & CS_{8,\mathcal{Q}_1} + CS_{8,\mathcal{Q}_2} + \frac{8\,\Delta}{n}\Big(- n_2\, CS_{7,\mathcal{Q}_1} + n_1\, CS_{7,\mathcal{Q}_2}\Big) \\ & + 28\Big(\frac{\Delta}{n}\Big)^2\Big({n_2}^2\, CS_{6,\mathcal{Q}_1} + {n_1}^2\, CS_{6,\mathcal{Q}_2}\Big) + 56\Big(\frac{\Delta}{n}\Big)^3\Big(- {n_2}^3\, CS_{5,\mathcal{Q}_1} + {n_1}^3\, CS_{5,\mathcal{Q}_2}\Big) \\ & + 70\Big(\frac{\Delta}{n}\Big)^4\Big({n_2}^4\, CS_{4,\mathcal{Q}_1} + {n_1}^4\, CS_{4,\mathcal{Q}_2}\Big) + 56\Big(\frac{\Delta}{n}\Big)^5\Big(- {n_2}^5\, CS_{3,\mathcal{Q}_1} + {n_1}^5\, CS_{3,\mathcal{Q}_2}\Big) \\ & + 28\Big(\frac{\Delta}{n}\Big)^6\Big({n_2}^6\, CS_{2,\mathcal{Q}_1} + {n_1}^6\, CS_{2,\mathcal{Q}_2}\Big) + n_1\, n_2\Big(\frac{\Delta}{n}\Big)^8\Big({n_1}^7 + {n_2}^7\Big)\end{aligned} \tag{12.8}$$

$$\begin{aligned}CS_{9,\mathcal{Q}} = {} & CS_{9,\mathcal{Q}_1} + CS_{9,\mathcal{Q}_2} + \frac{9\,\Delta}{n}\Big(- n_2\, CS_{8,\mathcal{Q}_1} + n_1\, CS_{8,\mathcal{Q}_2}\Big) \\ & + 36\Big(\frac{\Delta}{n}\Big)^2\Big({n_2}^2\, CS_{7,\mathcal{Q}_1} + {n_1}^2\, CS_{7,\mathcal{Q}_2}\Big) + 84\Big(\frac{\Delta}{n}\Big)^3\Big(- {n_2}^3\, CS_{6,\mathcal{Q}_1} + {n_1}^3\, CS_{6,\mathcal{Q}_2}\Big) \\ & + 126\Big(\frac{\Delta}{n}\Big)^4\Big({n_2}^4\, CS_{5,\mathcal{Q}_1} + {n_1}^4\, CS_{5,\mathcal{Q}_2}\Big) + 126\Big(\frac{\Delta}{n}\Big)^5\Big(- {n_2}^5\, CS_{4,\mathcal{Q}_1} + {n_1}^5\, CS_{4,\mathcal{Q}_2}\Big) \\ & + 84\Big(\frac{\Delta}{n}\Big)^6\Big({n_2}^6\, CS_{3,\mathcal{Q}_1} + {n_1}^6\, CS_{3,\mathcal{Q}_2}\Big) + 36\Big(\frac{\Delta}{n}\Big)^7\Big(- {n_2}^7\, CS_{2,\mathcal{Q}_1} + {n_1}^7\, CS_{2,\mathcal{Q}_2}\Big) \\ & + n_1\, n_2\Big(\frac{\Delta}{n}\Big)^9\Big({n_1}^8 - {n_2}^8\Big)\end{aligned} \tag{12.9}$$

$$\begin{aligned}CS_{10,\mathcal{Q}} = {} & CS_{10,\mathcal{Q}_1} + CS_{10,\mathcal{Q}_2} + \frac{10\,\Delta}{n}\Big(- n_2\, CS_{9,\mathcal{Q}_1} + n_1\, CS_{9,\mathcal{Q}_2}\Big) \\ & + 45\Big(\frac{\Delta}{n}\Big)^2\Big({n_2}^2\, CS_{8,\mathcal{Q}_1} + {n_1}^2\, CS_{8,\mathcal{Q}_2}\Big) + 120\Big(\frac{\Delta}{n}\Big)^3\Big(- {n_2}^3\, CS_{7,\mathcal{Q}_1} + {n_1}^3\, CS_{7,\mathcal{Q}_2}\Big) \\ & + 210\Big(\frac{\Delta}{n}\Big)^4\Big({n_2}^4\, CS_{6,\mathcal{Q}_1} + {n_1}^4\, CS_{6,\mathcal{Q}_2}\Big) + 252\Big(\frac{\Delta}{n}\Big)^5\Big(- {n_2}^5\, CS_{5,\mathcal{Q}_1} + {n_1}^5\, CS_{5,\mathcal{Q}_2}\Big) \\ & + 210\Big(\frac{\Delta}{n}\Big)^6\Big({n_2}^6\, CS_{4,\mathcal{Q}_1} + {n_1}^6\, CS_{4,\mathcal{Q}_2}\Big) + 120\Big(\frac{\Delta}{n}\Big)^7\Big(- {n_2}^7\, CS_{3,\mathcal{Q}_1} + {n_1}^7\, CS_{3,\mathcal{Q}_2}\Big) \\ & + 45\Big(\frac{\Delta}{n}\Big)^8\Big({n_2}^8\, CS_{2,\mathcal{Q}_1} + {n_1}^8\, CS_{2,\mathcal{Q}_2}\Big) + n_1\, n_2\Big(\frac{\Delta}{n}\Big)^{10}\Big({n_1}^9 + {n_2}^9\Big)\end{aligned} \tag{12.10}$$

12.1.2 Adjusted Central Sums

$$ACS_{1,\mathcal{Q}} = ACS_{1,\mathcal{Q}_1} + ACS_{1,\mathcal{Q}_2} + \frac{n_2\,(n_1)^2 + n_1\,(n_2)^2}{n^2}\Delta_t\Delta_l \tag{12.11}$$

$$\begin{aligned}ACS_{2,\mathcal{Q}} = {} & ACS_{2,\mathcal{Q}_1} + ACS_{2,\mathcal{Q}_2} + \frac{\Delta_l}{n}\left(n_1 CS_{2,\mathcal{Q}_2} - n_2 CS_{2,\mathcal{Q}_1}\right) \\ & + 2\left(\frac{\Delta_t}{n}\right)\left(n_1 ACS_{1,\mathcal{Q}_2} - n_2 ACS_{1,\mathcal{Q}_1}\right) + \frac{n_2\,(n_1)^3 - n_1\,(n_2)^3}{n^3}\left(\Delta_t\right)^2\Delta_l\end{aligned} \tag{12.12}$$

$$\begin{aligned}
ACS_{3,\mathcal{Q}} = {} & ACS_{3,\mathcal{Q}_1} + ACS_{3,\mathcal{Q}_2} + \frac{\Delta_l}{n}\left(n_1 CS_{3,\mathcal{Q}_2} - n_2 CS_{3,\mathcal{Q}_1}\right) \\
& + 3\left(\frac{\Delta_t}{n}\right)\left(n_1 ACS_{2,\mathcal{Q}_2} - n_2 ACS_{2,\mathcal{Q}_1} + \frac{\Delta_l}{n}\left(\left(n_1\right)^2 CS_{2,\mathcal{Q}_2} + \left(n_2\right)^2 CS_{2,\mathcal{Q}_1}\right)\right) \\
& + 3\left(\frac{\Delta_t}{n}\right)^2\left(\left(n_1\right)^2 ACS_{1,\mathcal{Q}_2} + \left(n_2\right)^2 ACS_{1,\mathcal{Q}_1}\right) + \frac{n_2\left(n_1\right)^4 + n_1\left(n_2\right)^4}{n^4}\left(\Delta_t\right)^3\Delta_l
\end{aligned} \tag{12.13}$$

$$\begin{aligned}
ACS_{4,\mathcal{Q}} = {} & ACS_{4,\mathcal{Q}_1} + ACS_{4,\mathcal{Q}_2} + \frac{\Delta_l}{n}\left(n_1 CS_{4,\mathcal{Q}_2} - n_2 CS_{4,\mathcal{Q}_1}\right) \\
& + 4\left(\frac{\Delta_t}{n}\right)\left(n_1 ACS_{3,\mathcal{Q}_2} - n_2 ACS_{3,\mathcal{Q}_1} + \frac{\Delta_l}{n}\left(\left(n_1\right)^2 CS_{3,\mathcal{Q}_2} + \left(n_2\right)^2 CS_{3,\mathcal{Q}_1}\right)\right) \\
& + 6\left(\frac{\Delta_t}{n}\right)^2\left(\left(n_1\right)^2 ACS_{2,\mathcal{Q}_2} + \left(n_2\right)^2 ACS_{2,\mathcal{Q}_1} + \frac{\Delta_l}{n}\left(\left(n_1\right)^3 CS_{2,\mathcal{Q}_2} - \left(n_2\right)^3 CS_{2,\mathcal{Q}_1}\right)\right) \\
& + 4\left(\frac{\Delta_t}{n}\right)^3\left(\left(n_1\right)^3 ACS_{1,\mathcal{Q}_2} - \left(n_2\right)^3 ACS_{1,\mathcal{Q}_1}\right) + \frac{n_2\left(n_1\right)^5 - n_1\left(n_2\right)^5}{n^5}\left(\Delta_t\right)^4\Delta_l
\end{aligned} \tag{12.14}$$

$$\begin{aligned}
ACS_{5,\mathcal{Q}} = {} & ACS_{5,\mathcal{Q}_1} + ACS_{5,\mathcal{Q}_2} + \frac{\Delta_l}{n}\left(n_1 CS_{5,\mathcal{Q}_2} - n_2 CS_{5,\mathcal{Q}_1}\right) \\
& + 5\left(\frac{\Delta_t}{n}\right)\left(n_1 ACS_{4,\mathcal{Q}_2} - n_2 ACS_{4,\mathcal{Q}_1} + \frac{\Delta_l}{n}\left(\left(n_1\right)^2 CS_{4,\mathcal{Q}_2} + \left(n_2\right)^2 CS_{4,\mathcal{Q}_1}\right)\right) \\
& + 10\left(\frac{\Delta_t}{n}\right)^2\left(\left(n_1\right)^2 ACS_{3,\mathcal{Q}_2} + \left(n_2\right)^2 ACS_{3,\mathcal{Q}_1} + \frac{\Delta_l}{n}\left(\left(n_1\right)^3 CS_{3,\mathcal{Q}_2} - \left(n_2\right)^3 CS_{3,\mathcal{Q}_1}\right)\right) \\
& + 10\left(\frac{\Delta_t}{n}\right)^3\left(\left(n_1\right)^3 ACS_{2,\mathcal{Q}_2} - \left(n_2\right)^3 ACS_{2,\mathcal{Q}_1} + \frac{\Delta_l}{n}\left(\left(n_1\right)^4 CS_{2,\mathcal{Q}_2} + \left(n_2\right)^4 CS_{2,\mathcal{Q}_1}\right)\right) \\
& + 5\left(\frac{\Delta_t}{n}\right)^4\left(\left(n_1\right)^4 ACS_{1,\mathcal{Q}_2} + \left(n_2\right)^4 ACS_{1,\mathcal{Q}_1}\right) + \frac{n_2\left(n_1\right)^6 + n_1\left(n_2\right)^6}{n^6}\left(\Delta_t\right)^5\Delta_l
\end{aligned} \tag{12.15}$$

12.2 Univariate Incremental

Below we consider $\Delta_t = t_n - \mu_{t,\mathcal{Q}_1}$ and $\Delta_l = l_n - \mu_{l,\mathcal{Q}_1}$ with $|\mathcal{Q}_1| = n - 1$. The extended set is defined as $\mathcal{Q} = \mathcal{Q}_1 \cup \left\{\, (t_n, l_n) \,\right\}$ with the cardinality of n. The computations related to the sample points should be repeated at each sample point independently.

12.2.1 Mean

$$\mu_{t,\mathcal{Q}} = \mu_{t,\mathcal{Q}_1} + \frac{\Delta_t}{n} \qquad \mu_{l,\mathcal{Q}} = \mu_{l,\mathcal{Q}_1} + \frac{\Delta_l}{n} \tag{12.16}$$

12.2.2 Central Sums

$$CS_{2,\mathcal{L}} = CS_{2,\mathcal{L}_1} + \frac{\left(\Delta_l\right)^2 (n-1)}{n} \tag{12.17}$$

$$CS_{2,\mathcal{Q}} = CS_{2,\mathcal{Q}_1} + \frac{\left(\Delta_t\right)^2 (n-1)}{n} \tag{12.18}$$

$$CS_{3,\mathcal{Q}} = CS_{3,\mathcal{Q}_1} - \frac{3\Delta_t}{n} CS_{2,\mathcal{Q}_1} + \frac{\left(\Delta_t\right)^3 (n-1)((n-1)^2 - 1)}{n^3} \tag{12.19}$$

$$CS_{4,\mathcal{Q}} = CS_{4,\mathcal{Q}_1} - \frac{4\Delta_t}{n} CS_{3,\mathcal{Q}_1} + \frac{6\left(\Delta_t\right)^2}{n^2} CS_{2,\mathcal{Q}_1} + \frac{\left(\Delta_t\right)^4 (n-1)((n-1)^3+1)}{n^4} \tag{12.20}$$

$$\begin{aligned} CS_{5,\mathcal{Q}} = {} & CS_{5,\mathcal{Q}_1} - \frac{5\Delta_t}{n} CS_{4,\mathcal{Q}_1} + \frac{10\left(\Delta_t\right)^2}{n^2} CS_{3,\mathcal{Q}_1} - \frac{10\left(\Delta_t\right)^3}{n^3} CS_{2,\mathcal{Q}_1} \\ & + \frac{\left(\Delta_t\right)^5 (n-1)((n-1)^4-1)}{n^5} \end{aligned} \tag{12.21}$$

$$\begin{aligned} CS_{6,\mathcal{Q}} = {} & CS_{6,\mathcal{Q}_1} - \frac{6\Delta_t}{n} CS_{5,\mathcal{Q}_1} + \frac{15\left(\Delta_t\right)^2}{n^2} CS_{4,\mathcal{Q}_1} - \frac{20\left(\Delta_t\right)^3}{n^3} CS_{3,\mathcal{Q}_1} \\ & + \frac{15\left(\Delta_t\right)^4}{n^4} CS_{2,\mathcal{Q}_1} + \frac{\left(\Delta_t\right)^6 (n-1)((n-1)^5+1)}{n^6} \end{aligned} \tag{12.22}$$

$$\begin{aligned} CS_{7,\mathcal{Q}} = {} & CS_{7,\mathcal{Q}_1} - \frac{7\Delta_t}{n} CS_{6,\mathcal{Q}_1} + \frac{21\left(\Delta_t\right)^2}{n^2} CS_{5,\mathcal{Q}_1} - \frac{35\left(\Delta_t\right)^3}{n^3} CS_{4,\mathcal{Q}_1} \\ & + \frac{35\left(\Delta_t\right)^4}{n^4} CS_{3,\mathcal{Q}_1} - \frac{21\left(\Delta_t\right)^5}{n^5} CS_{2,\mathcal{Q}_1} + \frac{\left(\Delta_t\right)^7 (n-1)((n-1)^6-1)}{n^7} \end{aligned} \tag{12.23}$$

$$\begin{aligned} CS_{8,\mathcal{Q}} = {} & CS_{8,\mathcal{Q}_1} - \frac{8\Delta_t}{n} CS_{7,\mathcal{Q}_1} + \frac{28\left(\Delta_t\right)^2}{n^2} CS_{6,\mathcal{Q}_1} - \frac{56\left(\Delta_t\right)^3}{n^3} CS_{5,\mathcal{Q}_1} \\ & + \frac{70\left(\Delta_t\right)^4}{n^4} CS_{4,\mathcal{Q}_1} - \frac{56\left(\Delta_t\right)^5}{n^5} CS_{3,\mathcal{Q}_1} + \frac{28\left(\Delta_t\right)^6}{n^6} CS_{2,\mathcal{Q}_1} \\ & + \frac{\left(\Delta_t\right)^8 (n-1)((n-1)^7+1)}{n^8} \end{aligned} \tag{12.24}$$

$$\begin{aligned} CS_{9,\mathcal{Q}} = {} & CS_{9,\mathcal{Q}_1} - \frac{9\Delta_t}{n} CS_{8,\mathcal{Q}_1} + \frac{36\left(\Delta_t\right)^2}{n^2} CS_{7,\mathcal{Q}_1} - \frac{84\left(\Delta_t\right)^3}{n^3} CS_{6,\mathcal{Q}_1} \\ & + \frac{126\left(\Delta_t\right)^4}{n^4} CS_{5,\mathcal{Q}_1} - \frac{126\left(\Delta_t\right)^5}{n^5} CS_{4,\mathcal{Q}_1} + \frac{84\left(\Delta_t\right)^6}{n^6} CS_{3,\mathcal{Q}_1} \\ & - \frac{36\left(\Delta_t\right)^7}{n^7} CS_{2,\mathcal{Q}_1} + \frac{\left(\Delta_t\right)^9 (n-1)((n-1)^8-1)}{n^9} \end{aligned} \tag{12.25}$$

$$\begin{aligned} CS_{10,\mathcal{Q}} = {} & CS_{10,\mathcal{Q}_1} - \frac{10\Delta_t}{n} CS_{9,\mathcal{Q}_1} + \frac{45\left(\Delta_t\right)^2}{n^2} CS_{8,\mathcal{Q}_1} - \frac{120\left(\Delta_t\right)^3}{n^3} CS_{7,\mathcal{Q}_1} \\ & + \frac{210\left(\Delta_t\right)^4}{n^4} CS_{6,\mathcal{Q}_1} - \frac{252\left(\Delta_t\right)^5}{n^5} CS_{5,\mathcal{Q}_1} + \frac{210\left(\Delta_t\right)^6}{n^6} CS_{4,\mathcal{Q}_1} \\ & - \frac{120\left(\Delta_t\right)^7}{n^7} CS_{3,\mathcal{Q}_1} + \frac{45\left(\Delta_t\right)^8}{n^8} CS_{2,\mathcal{Q}_1} + \frac{\left(\Delta_t\right)^{10} (n-1)((n-1)^9+1)}{n^{10}} \end{aligned} \tag{12.26}$$

12.2.3 Adjusted Central Sums

$$ACS_{1,\mathcal{Q}} = ACS_{1,\mathcal{Q}_1} + \frac{(n-1)^2+n-1}{n^2} \Delta_t \Delta_l \tag{12.27}$$

$$ACS_{2,\mathcal{Q}} = ACS_{2,\mathcal{Q}_1} - \frac{\Delta_l}{n} CS_{2,\mathcal{Q}_1} - 2\left(\frac{\Delta_t}{n}\right) ACS_{1,\mathcal{Q}_1} + \frac{(n-1)^3-n+1}{n^3} \left(\Delta_t\right)^2 \Delta_l \tag{12.28}$$

$$\begin{aligned} ACS_{3,\mathcal{Q}} = {} & ACS_{3,\mathcal{Q}_1} - \frac{\Delta_l}{n} CS_{3,\mathcal{Q}_1} - 3\left(\frac{\Delta_t}{n}\right)\left(ACS_{2,\mathcal{Q}_1} - \frac{\Delta_l}{n} CS_{2,\mathcal{Q}_1}\right) \\ & + 3\left(\frac{\Delta_t}{n}\right)^2 ACS_{1,\mathcal{Q}_1} + \frac{(n-1)^4 + n - 1}{n^4} \left(\Delta_t\right)^3 \Delta_l \end{aligned} \tag{12.29}$$

$$\begin{aligned} ACS_{4,\mathcal{Q}} = {} & ACS_{4,\mathcal{Q}_1} - \frac{\Delta_l}{n} CS_{4,\mathcal{Q}_1} - 4\left(\frac{\Delta_t}{n}\right)\left(ACS_{3,\mathcal{Q}_1} - \frac{\Delta_l}{n} CS_{3,\mathcal{Q}_1}\right) \\ & + 6\left(\frac{\Delta_t}{n}\right)^2 \left(ACS_{2,\mathcal{Q}_1} - \frac{\Delta_l}{n} CS_{2,\mathcal{Q}_1}\right) \\ & - 4\left(\frac{\Delta_t}{n}\right)^3 ACS_{1,\mathcal{Q}_1} + \frac{(n-1)^5 - n + 1}{n^5} \left(\Delta_t\right)^4 \Delta_l \end{aligned} \tag{12.30}$$

$$\begin{aligned} ACS_{5,\mathcal{Q}} = {} & ACS_{5,\mathcal{Q}_1} - \frac{\Delta_l}{n} CS_{5,\mathcal{Q}_1} - 5\left(\frac{\Delta_t}{n}\right)\left(ACS_{4,\mathcal{Q}_1} - \frac{\Delta_l}{n} CS_{4,\mathcal{Q}_1}\right) \\ & + 10\left(\frac{\Delta_t}{n}\right)^2 \left(ACS_{3,\mathcal{Q}_1} - \frac{\Delta_l}{n} CS_{3,\mathcal{Q}_1}\right) - 10\left(\frac{\Delta_t}{n}\right)^3 \left(ACS_{2,\mathcal{Q}_1} - \frac{\Delta_l}{n} CS_{2,\mathcal{Q}_1}\right) \\ & + 4\left(\frac{\Delta_t}{n}\right)^4 ACS_{1,\mathcal{Q}_1} + \frac{(n-1)^6 + n - 1}{n^6} \left(\Delta_t\right)^5 \Delta_l \end{aligned} \tag{12.31}$$

12.3 Central Moments from the Raw Moments

$$CM_2 = M_2 - {M_1}^2 \tag{12.32}$$

$$CM_3 = M_3 - 3\, M_2\, M_1 + 2\, {M_1}^3 \tag{12.33}$$

$$CM_4 = M_4 - 4\, M_3\, M_1 + 6\, M_2\, {M_1}^2 - 3\, {M_1}^4 \tag{12.34}$$

$$CM_5 = M_5 - 5\, M_4\, M_1 + 10\, M_3\, {M_1}^2 - 10\, M_2\, {M_1}^3 + 4\, {M_1}^5 \tag{12.35}$$

$$CM_6 = M_6 - 6\, M_5\, M_1 + 15\, M_4\, {M_1}^2 - 20\, M_3\, {M_1}^3 + 15\, M_2\, {M_1}^4 - 5\, {M_1}^6 \tag{12.36}$$

$$CM_7 = M_7 - 7\, M_6\, M_1 + 21\, M_5\, {M_1}^2 - 35\, M_4\, {M_1}^3 + 35\, M_3\, {M_1}^4 - 21\, M_2\, {M_1}^5 + 6\, {M_1}^7 \tag{12.37}$$

$$\begin{aligned} CM_8 = {} & M_8 - 8\, M_7\, M_1 + 28\, M_6\, {M_1}^2 - 56\, M_5\, {M_1}^3 + 70\, M_4\, {M_1}^4 - 56\, M_3\, {M_1}^5 \\ & + 28\, M_2\, {M_1}^6 - 7\, {M_1}^8 \end{aligned} \tag{12.38}$$

$$\begin{aligned} CM_9 = {} & M_9 - 9\, M_8\, M_1 + 36\, M_7\, {M_1}^2 - 84\, M_6\, {M_1}^3 + 126\, M_5\, {M_1}^4 - 126\, M_4\, {M_1}^5 \\ & + 84\, M_3\, {M_1}^6 - 36\, M_2\, {M_1}^7 + 8\, {M_1}^9 \end{aligned} \tag{12.39}$$

$$\begin{aligned} CM_{10} = {} & M_{10} - 10\, M_9\, M_1 + 45\, M_8\, {M_1}^2 - 120\, M_7\, {M_1}^3 + 210\, M_6\, {M_1}^4 - 252\, M_5\, {M_1}^5 \\ & + 210\, M_4\, {M_1}^6 - 120\, M_3\, {M_1}^7 + 45\, M_2\, {M_1}^8 - 9\, {M_1}^{10} \end{aligned} \tag{12.40}$$

12.4 Mean and Variance of the Preprocessed Measurements

$$\text{(1st order)} \quad \mu = M_1, \qquad \sigma^2 = CM_2 \tag{12.41}$$

$$\text{(2nd order)} \quad \mu = CM_2, \qquad \sigma^2 = CM_4 - {CM_2}^2 \tag{12.42}$$

$$\text{(3rd order)} \quad \mu = SM_3 = \frac{CM_3}{\left(\sqrt{CM_2}\right)^3}, \qquad \sigma^2 = \frac{CM_6 - {CM_3}^2}{{CM_2}^3} \tag{12.43}$$

$$\text{(4th order)} \quad \mu = SM_4 = \frac{CM_4}{\left(\sqrt{CM_2}\right)^4}, \qquad \sigma^2 = \frac{CM_8 - {CM_4}^2}{{CM_2}^4} \tag{12.44}$$

$$\text{(5th order)} \quad \mu = SM_5 = \frac{CM_5}{\left(\sqrt{CM_2}\right)^5}, \qquad \sigma^2 = \frac{CM_{10} - {CM_5}^2}{{CM_2}^5} \tag{12.45}$$

12.5 Univariate Correlation

The correlation of a univariate CPA can be estimated as

$$\text{(1st order)} \quad \rho = \frac{\frac{1}{n} ACS_1}{\sqrt{\frac{1}{n} CS_{2,\mathcal{Q}} \, \frac{1}{n} CS_{2,\mathcal{L}}}} \tag{12.46}$$

$$\text{(2nd order)} \quad \rho = \frac{\frac{1}{n} ACS_2}{\sqrt{\frac{1}{n}\left(CS_{4,\mathcal{Q}} - \frac{\left(CS_{2,\mathcal{Q}}\right)^2}{n}\right) \frac{1}{n} CS_{2,\mathcal{L}}}} \tag{12.47}$$

$$\text{(3rd order)} \quad \rho = \frac{\frac{1}{n} ACS_3}{\sqrt{\frac{1}{n}\left(CS_{6,\mathcal{Q}} - \frac{\left(CS_{3,\mathcal{Q}}\right)^2}{n}\right) \frac{1}{n} CS_{2,\mathcal{L}}}} \tag{12.48}$$

$$\text{(4th order)} \quad \rho = \frac{\frac{1}{n} ACS_4}{\sqrt{\frac{1}{n}\left(CS_{8,\mathcal{Q}} - \frac{\left(CS_{4,\mathcal{Q}}\right)^2}{n}\right) \frac{1}{n} CS_{2,\mathcal{L}}}} \tag{12.49}$$

$$\text{(5th order)} \quad \rho = \frac{\frac{1}{n} ACS_5}{\sqrt{\frac{1}{n}\left(CS_{10,\mathcal{Q}} - \frac{\left(CS_{5,\mathcal{Q}}\right)^2}{n}\right) \frac{1}{n} CS_{2,\mathcal{L}}}} \tag{12.50}$$

12.6 Bivariate Second-Order Evaluation

In the following we give the necessary formulas for a bivariate second-order CPA or t-test for exemplary sample points $\mathcal{J} = \{1, 2\}$. Further, we consider $\mathcal{J}' = \{1, 2, *\}$ with $t_i^{(*)} = l_i$ and $t_i^{(*)} = l_i$. For the computation of the denominator we also consider $\mathcal{J}'' = \{1, 2, 1, 2\}$.

12.6.1 Two-Pair Iterative

We consider two sets $\mathcal{Q}_1$, $\mathcal{Q}_2$ with $|\mathcal{Q}_1| = n_1$, $|\mathcal{Q}_2| = n_2$ and $\Delta^{(j \in \mathcal{J})} = \mu_{\mathcal{Q}_2}^{(j)} - \mu_{\mathcal{Q}_1}^{(j)}$, with $\mu_{\mathcal{Q}_i}^{(j)}$ as the mean of the set $\mathcal{Q}_i$ at sample point j. The two sets make up the extended set $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$ which has a cardinality of n.

Sum of Centered Products (Numerator)

$$SCP_{2,\mathcal{Q},\{1,2\}} = SCP_{2,\mathcal{Q}_1,\{1,2\}} + SCP_{2,\mathcal{Q}_2,\{1,2\}} + \frac{\left((n_1)^2 n_2 + n_1 (n_2)^2\right) \Delta^{(1)}\Delta^{(2)}}{n^2} \tag{12.51}$$

$$SCP_{2,\mathcal{Q},\{1,*\}} = SCP_{2,\mathcal{Q}_1,\{1,*\}} + SCP_{2,\mathcal{Q}_2,\{1,*\}} + \frac{\left((n_1)^2 n_2 + n_1 (n_2)^2\right) \Delta^{(1)}\Delta^{(*)}}{n^2} \tag{12.52}$$

$$SCP_{2,\mathcal{Q},\{2,*\}} = SCP_{2,\mathcal{Q}_1,\{2,*\}} + SCP_{2,\mathcal{Q}_2,\{2,*\}} + \frac{\left((n_1)^2 n_2 + n_1 (n_2)^2\right) \Delta^{(2)}\Delta^{(*)}}{n^2} \tag{12.53}$$

$$\begin{aligned} SCP_{3,\mathcal{Q},\{1,2,*\}} = {} & SCP_{3,\mathcal{Q}_1,\{1,2,*\}} + SCP_{3,\mathcal{Q}_2,\{1,2,*\}} \\ & + \left(n_1 SCP_{2,\mathcal{Q}_2,\{1,2\}} - n_2 SCP_{2,\mathcal{Q}_1,\{1,2\}}\right) \frac{\Delta^{(*)}}{n} \\ & + \left(n_1 SCP_{2,\mathcal{Q}_2,\{1,*\}} - n_2 SCP_{2,\mathcal{Q}_1,\{1,*\}}\right) \frac{\Delta^{(2)}}{n} \\ & + \left(n_1 SCP_{2,\mathcal{Q}_2,\{2,*\}} - n_2 SCP_{2,\mathcal{Q}_1,\{2,*\}}\right) \frac{\Delta^{(1)}}{n} \\ & + \frac{\left((n_1)^3 n_2 - n_1 (n_2)^3\right) \Delta^{(1)}\Delta^{(2)}\Delta^{(*)}}{n^3} \end{aligned} \tag{12.54}$$

Sum of Centered Products (Denominator)

$$SCP_{2,\mathcal{Q},\{1,1\}} = SCP_{2,\mathcal{Q}_1,\{1,1\}} + SCP_{2,\mathcal{Q}_2,\{1,1\}} + \frac{\left((n_1)^2 n_2 + n_1 (n_2)^2\right) \Delta^{(1)}\Delta^{(1)}}{n^2} \tag{12.55}$$

$$SCP_{2,\mathcal{Q},\{1,2\}} = SCP_{2,\mathcal{Q}_1,\{1,2\}} + SCP_{2,\mathcal{Q}_2,\{1,2\}} + \frac{\left((n_1)^2 n_2 + n_1 (n_2)^2\right) \Delta^{(1)}\Delta^{(2)}}{n^2} \tag{12.56}$$

$$SCP_{2,\mathcal{Q},\{2,2\}} = SCP_{2,\mathcal{Q}_1,\{2,2\}} + SCP_{2,\mathcal{Q}_2,\{2,2\}} + \frac{\left((n_1)^2 n_2 + n_1 (n_2)^2\right) \Delta^{(2)}\Delta^{(2)}}{n^2} \tag{12.57}$$

$$\begin{aligned} SCP_{3,\mathcal{Q},\{1,2,1\}} = {} & SCP_{3,\mathcal{Q}_1,\{1,2,1\}} + SCP_{3,\mathcal{Q}_2,\{1,2,1\}} \\ & + \left(n_1 SCP_{2,\mathcal{Q}_2,\{1,2\}} - n_2 SCP_{2,\mathcal{Q}_1,\{1,2\}}\right) \frac{2\Delta^{(1)}}{n} \\ & + \left(n_1 SCP_{2,\mathcal{Q}_2,\{1,1\}} - n_2 SCP_{2,\mathcal{Q}_1,\{1,1\}}\right) \frac{\Delta^{(2)}}{n} \\ & + \frac{\left((n_1)^3 n_2 - n_1 (n_2)^3\right) \Delta^{(1)}\Delta^{(2)}\Delta^{(1)}}{n^3} \end{aligned} \tag{12.58}$$

$$\begin{aligned}
SCP_{3,\mathcal{Q},\{1,2,2\}} = {} & SCP_{3,\mathcal{Q}_1,\{1,2,2\}} + SCP_{3,\mathcal{Q}_2,\{1,2,2\}} \\
& + \left(n_1 SCP_{2,\mathcal{Q}_2,\{1,2\}} - n_2 SCP_{2,\mathcal{Q}_1,\{1,2\}}\right) \frac{2\Delta^{(2)}}{n} \\
& + \left(n_1 SCP_{2,\mathcal{Q}_2,\{2,2\}} - n_2 SCP_{2,\mathcal{Q}_1,\{2,2\}}\right) \frac{\Delta^{(1)}}{n} \\
& + \frac{\left((n_1)^3 n_2 - n_1 (n_2)^3\right) \Delta^{(1)}\Delta^{(2)}\Delta^{(2)}}{n^3}
\end{aligned} \tag{12.59}$$

$$\begin{aligned}
SCP_{4,\mathcal{Q},\{1,2,1,2\}} = {} & SCP_{4,\mathcal{Q}_1,\{1,2,1,2\}} + SCP_{4,\mathcal{Q}_2,\{1,2,1,2\}} \\
& + \left(n_1 SCP_{3,\mathcal{Q}_2,\{1,2,1\}} - n_2 SCP_{3,\mathcal{Q}_1,\{1,2,1\}}\right) \frac{2\Delta^{(2)}}{n} \\
& + \left(n_1 SCP_{3,\mathcal{Q}_2,\{1,2,2\}} - n_2 SCP_{3,\mathcal{Q}_1,\{1,2,2\}}\right) \frac{2\Delta^{(1)}}{n} \\
& + \left((n_1)^2 SCP_{2,\mathcal{Q}_2,\{1,2\}} + (n_2)^2 SCP_{2,\mathcal{Q}_1,\{1,2\}}\right) \frac{2\Delta^{(1)}\Delta^{(2)}}{n^2} \\
& + \left((n_1)^2 SCP_{2,\mathcal{Q}_2,\{1,1\}} + (n_2)^2 SCP_{2,\mathcal{Q}_1,\{1,1\}}\right) \frac{\Delta^{(2)}\Delta^{(2)}}{n^2} \\
& + \left((n_1)^2 SCP_{2,\mathcal{Q}_2,\{2,2\}} + (n_2)^2 SCP_{2,\mathcal{Q}_1,\{2,2\}}\right) \frac{\Delta^{(1)}\Delta^{(1)}}{n^2} \\
& + \frac{\left((n_1)^4 n_2 + n_1 (n_2)^4\right) \Delta^{(1)}\Delta^{(2)}\Delta^{(1)}\Delta^{(2)}}{n^4}
\end{aligned} \tag{12.60}$$

12.6.2 Incremental

Below we consider $\Delta^{(j\in\mathcal{J})} = t_n^{(j)} - \mu_{\mathcal{Q}_1}^{(j)}$ with $|\mathcal{Q}_1| = n - 1$. The extended set is defined as $\mathcal{Q} = \mathcal{Q}_1 \cup \left\{(t_n^{(j)} | \forall j \in \mathcal{J})\right\}$ with the cardinality of n.

Sum of Centered Products (Numerator)

$$SCP_{2,\mathcal{Q},\{1,2\}} = SCP_{2,\mathcal{Q}_1,\{1,2\}} + \frac{(n-1)\Delta^{(1)}\Delta^{(2)}}{n} \tag{12.61}$$

$$SCP_{2,\mathcal{Q},\{1,*\}} = SCP_{2,\mathcal{Q}_1,\{1,*\}} + \frac{(n-1)\Delta^{(1)}\Delta^{(*)}}{n} \tag{12.62}$$

$$SCP_{2,\mathcal{Q},\{2,*\}} = SCP_{2,\mathcal{Q}_1,\{2,*\}} + \frac{(n-1)\Delta^{(2)}\Delta^{(*)}}{n} \tag{12.63}$$

$$\begin{aligned}
SCP_{3,\mathcal{Q},\{1,2,*\}} = {} & SCP_{3,\mathcal{Q}_1,\{1,2,*\}} - SCP_{2,\mathcal{Q}_1,\{1,2\}} \frac{\Delta^{(*)}}{n} - SCP_{2,\mathcal{Q}_1,\{1,*\}} \frac{\Delta^{(2)}}{n} \\
& - SCP_{2,\mathcal{Q}_1,\{2,*\}} \frac{\Delta^{(1)}}{n} + \frac{\left(n^2 - 3n + 2\right) \Delta^{(1)}\Delta^{(2)}\Delta^{(*)}}{n^2}
\end{aligned} \tag{12.64}$$

Sum of Centered Products (Denominator)

$$SCP_{2,\mathcal{Q},\{1,1\}} = SCP_{2,\mathcal{Q}_1,\{1,1\}} + \frac{(n-1)\Delta^{(1)}\Delta^{(1)}}{n} \tag{12.65}$$

$$
\begin{aligned}
SCP_{2,\mathcal{Q},\{1,2\}} &= SCP_{2,\mathcal{Q}_1,\{1,2\}} + \frac{(n-1)\Delta^{(1)}\Delta^{(2)}}{n} && (12.66)\\
SCP_{2,\mathcal{Q},\{2,2\}} &= SCP_{2,\mathcal{Q}_1,\{2,2\}} + \frac{(n-1)\Delta^{(2)}\Delta^{(2)}}{n} && (12.67)\\
SCP_{3,\mathcal{Q},\{1,2,1\}} &= SCP_{3,\mathcal{Q}_1,\{1,2,1\}} - 2\,SCP_{2,\mathcal{Q}_1,\{1,2\}}\frac{\Delta^{(1)}}{n} - SCP_{2,\mathcal{Q}_1,\{1,1\}}\frac{\Delta^{(2)}}{n} && (12.68)\\
&\quad + \frac{\Delta^{(1)}\Delta^{(2)}\Delta^{(1)}\left(n^2-3n+2\right)}{n^2}\\
SCP_{3,\mathcal{Q},\{1,2,2\}} &= SCP_{3,\mathcal{Q}_1,\{1,2,2\}} - 2\,SCP_{2,\mathcal{Q}_1,\{1,2\}}\frac{\Delta^{(2)}}{n} - SCP_{2,\mathcal{Q}_1,\{2,2\}}\frac{\Delta^{(1)}}{n} && (12.69)\\
&\quad + \frac{\left(n^2-3n+2\right)\Delta^{(1)}\Delta^{(2)}\Delta^{(2)}}{n^2}\\
SCP_{4,\mathcal{Q},\{1,2,1,2\}} &= SCP_{4,\mathcal{Q}_1,\{1,2,1,2\}} - 2\,SCP_{3,\mathcal{Q}_1,\{1,2,1\}}\frac{\Delta^{(2)}}{n} - 2\,SCP_{3,\mathcal{Q}_1,\{1,2,2\}}\frac{\Delta^{(1)}}{n} && (12.70)\\
&\quad + SCP_{2,\mathcal{Q}_1,\{1,1\}}\frac{\Delta^{(2)}\Delta^{(2)}}{n^2} + 4\,SCP_{2,\mathcal{Q}_1,\{1,2\}}\frac{\Delta^{(1)}\Delta^{(2)}}{n^2}\\
&\quad + SCP_{2,\mathcal{Q}_1,\{2,2\}}\frac{\Delta^{(1)}\Delta^{(1)}}{n^2} + \frac{\left(n^3-4n^2+6n-3\right)\Delta^{(1)}\Delta^{(2)}\Delta^{(1)}\Delta^{(2)}}{n^3}
\end{aligned}
$$

At any time the correlation of a bivariate CPA can be estimated as

$$
\rho = \frac{\frac{1}{n}\,SCP_{3,\mathcal{Q},\{1,2,*\}}}{\sqrt{\frac{1}{n}\,SCP_{4,\mathcal{Q},\{1,2,1,2\}}\,\frac{1}{n}\,CS_{2,\mathcal{L}}}}, \tag{12.71}
$$

where the central sum $CS_{2,\mathcal{L}}$ is derived from Equation (12.1) or Equation (12.17).

In this scenario, $\mu = \frac{C_{2,\mathcal{Q}',\{1,2\}}}{n}$ corresponds to the first parameter and $s^2 = \frac{C_{4,\mathcal{Q}',\{1,2,1,2\}}}{n} - \mu^2$ to the second parameter of a bivariate second-order t-test.

12.6.3 Correlation from the Raw Moments

12.6.4 Third Order

$$
\begin{aligned}
\lambda_1 &= S_3^{(t,l)} - 3\frac{S_1^{(t)}S_2^{(t,l)}}{n} + 3\frac{\left(S_1^{(t)}\right)^2 S_1^{(t,l)}}{n^2} - \frac{\left(S_1^{(t)}\right)^3 S_1^{(l)}}{n^3} && (12.72)\\
\lambda_2 &= S_3^{(t)} - 3\frac{S_1^{(t)}S_2^{(t)}}{n} + 2\frac{\left(S_1^{(t)}\right)^3}{n^2} && (12.73)\\
\lambda_3 &= S_6^{(t)} - 6\frac{S_1^{(t)}S_5^{(t)}}{n} + 15\frac{\left(S_1^{(t)}\right)^2 S_4^{(t)}}{n^2} - 20\frac{\left(S_1^{(t)}\right)^3 S_3^{(t)}}{n^3} + 15\frac{\left(S_1^{(t)}\right)^4 S_2^{(t)}}{n^4} - 5\frac{\left(S_1^{(t)}\right)^6}{n^5} && (12.74)
\end{aligned}
$$

12.6.5 Fourth Order

$$\lambda_1 = S_4^{(t,l)} - 4\frac{S_1^{(t)} S_3^{(t,l)}}{n} + 6\frac{\left(S_1^{(t)}\right)^2 S_2^{(t,l)}}{n^2} - 4\frac{\left(S_1^{(t)}\right)^3 S_1^{(t,l)}}{n^3} + \frac{\left(S_1^{(t)}\right)^4 S_1^{(l)}}{n^4} \tag{12.75}$$

$$\lambda_2 = S_4^{(t)} - 4\frac{S_1^{(t)} S_3^{(t)}}{n} + 6\frac{\left(S_1^{(t)}\right)^2 S_2^{(t)}}{n^2} - 3\frac{\left(S_1^{(t)}\right)^4}{n^3} \tag{12.76}$$

$$\begin{aligned}\lambda_3 = S_8^{(t)} &- 8\frac{S_1^{(t)} S_7^{(t)}}{n} + 28\frac{\left(S_1^{(t)}\right)^2 S_6^{(t)}}{n^2} - 56\frac{\left(S_1^{(t)}\right)^3 S_5^{(t)}}{n^3} + 70\frac{\left(S_1^{(t)}\right)^4 S_4^{(t)}}{n^4} \\ &- 56\frac{\left(S_1^{(t)}\right)^5 S_3^{(t)}}{n^5} + 28\frac{\left(S_1^{(t)}\right)^6 S_2^{(t)}}{n^6} - 7\frac{\left(S_1^{(t)}\right)^8}{n^7}\end{aligned} \tag{12.77}$$

12.6.6 Fifth Order

$$\begin{aligned}\lambda_1 = S_5^{(t,l)} &- 5\frac{S_1^{(t)} S_4^{(t,l)}}{n} + 10\frac{\left(S_1^{(t)}\right)^2 S_3^{(t,l)}}{n^2} - 10\frac{\left(S_1^{(t)}\right)^3 S_2^{(t,l)}}{n^3} + 5\frac{\left(S_1^{(t)}\right)^4 S_1^{(t,l)}}{n^4} \\ &- \frac{\left(S_1^{(t)}\right)^5 S_1^{(l)}}{n^5}\end{aligned} \tag{12.78}$$

$$\lambda_2 = S_5^{(t)} - 5\frac{S_1^{(t)} S_4^{(t)}}{n} + 10\frac{\left(S_1^{(t)}\right)^2 S_3^{(t)}}{n^2} - 10\frac{\left(S_1^{(t)}\right)^3 S_2^{(t)}}{n^3} + 4\frac{\left(S_1^{(t)}\right)^5}{n^4} \tag{12.79}$$

$$\begin{aligned}\lambda_3 = S_{10}^{(t)} &- 10\frac{S_1^{(t)} S_9^{(t)}}{n} + 45\frac{\left(S_1^{(t)}\right)^2 S_8^{(t)}}{n^2} - 120\frac{\left(S_1^{(t)}\right)^3 S_7^{(t)}}{n^3} + 210\frac{\left(S_1^{(t)}\right)^4 S_6^{(t)}}{n^4} \\ &- 252\frac{\left(S_1^{(t)}\right)^5 S_5^{(t)}}{n^5} + 210\frac{\left(S_1^{(t)}\right)^6 S_4^{(t)}}{n^6} - 120\frac{\left(S_1^{(t)}\right)^7 S_3^{(t)}}{n^7} + 45\frac{\left(S_1^{(t)}\right)^8 S_2^{(t)}}{n^8} - 9\frac{\left(S_1^{(t)}\right)^{10}}{n^9}\end{aligned} \tag{12.80}$$

Chapter 13

Second-Order Threshold Implementation of RCA and KSA

13.1 Second-Order RCA

13.1.1 Carry (1. Step)

$$\widetilde{c}_{i\oplus 1}^{1} = a_i^2 b_i^2 \oplus a_i^1 b_i^2 \oplus a_i^2 b_i^1 \oplus a_i^2 c_i^2 \oplus a_i^1 c_i^2 \oplus a_i^2 c_i^1 \oplus c_i^2 b_i^2 \oplus c_i^1 b_i^2 \oplus c_i^2 b_i^1 \tag{13.1}$$
$$\widetilde{c}_{i\oplus 1}^{2} = a_i^3 b_i^3 \oplus a_i^1 b_i^3 \oplus a_i^3 b_i^1 \oplus a_i^3 c_i^3 \oplus a_i^1 c_i^3 \oplus a_i^3 c_i^1 \oplus c_i^3 b_i^3 \oplus c_i^1 b_i^3 \oplus c_i^3 b_i^1 \tag{13.2}$$
$$\widetilde{c}_{i\oplus 1}^{3} = a_i^4 b_i^4 \oplus a_i^1 b_i^4 \oplus a_i^4 b_i^1 \oplus a_i^4 c_i^4 \oplus a_i^1 c_i^4 \oplus a_i^4 c_i^1 \oplus c_i^4 b_i^4 \oplus c_i^1 b_i^4 \oplus c_i^4 b_i^1 \tag{13.3}$$
$$\widetilde{c}_{i\oplus 1}^{4} = a_i^1 b_i^1 \oplus a_i^1 b_i^5 \oplus a_i^5 b_i^1 \oplus a_i^1 c_i^1 \oplus a_i^1 c_i^5 \oplus a_i^5 c_i^1 \oplus c_i^1 b_i^1 \oplus c_i^1 b_i^5 \oplus c_i^5 b_i^1 \tag{13.4}$$
$$\widetilde{c}_{i\oplus 1}^{5} = a_i^2 b_i^3 \oplus a_i^3 b_i^2 \oplus a_i^2 c_i^3 \oplus a_i^3 c_i^2 \oplus c_i^2 b_i^3 \oplus c_i^3 b_i^2 \tag{13.5}$$
$$\widetilde{c}_{i\oplus 1}^{6} = a_i^2 b_i^4 \oplus a_i^4 b_i^2 \oplus a_i^2 c_i^4 \oplus a_i^4 c_i^2 \oplus c_i^2 b_i^4 \oplus c_i^4 b_i^2 \tag{13.6}$$
$$\widetilde{c}_{i\oplus 1}^{7} = a_i^5 b_i^5 \oplus a_i^2 b_i^5 \oplus a_i^5 b_i^2 \oplus a_i^5 c_i^5 \oplus a_i^2 c_i^5 \oplus a_i^5 c_i^2 \oplus c_i^5 b_i^5 \oplus c_i^2 b_i^5 \oplus c_i^5 b_i^2 \tag{13.7}$$
$$\widetilde{c}_{i\oplus 1}^{8} = a_i^3 b_i^4 \oplus a_i^4 b_i^3 \oplus a_i^3 c_i^4 \oplus a_i^4 c_i^3 \oplus c_i^3 b_i^4 \oplus c_i^4 b_i^3 \tag{13.8}$$
$$\widetilde{c}_{i\oplus 1}^{9} = a_i^3 b_i^5 \oplus a_i^5 b_i^3 \oplus a_i^3 c_i^5 \oplus a_i^5 c_i^3 \oplus c_i^3 b_i^5 \oplus c_i^5 b_i^3 \tag{13.9}$$
$$\widetilde{c}_{i\oplus 1}^{10} = a_i^4 b_i^5 \oplus a_i^5 b_i^4 \oplus a_i^4 c_i^5 \oplus a_i^5 c_i^4 \oplus c_i^4 b_i^5 \oplus c_i^5 b_i^4 \tag{13.10}$$

13.1.2 Carry (2. Step)

$$c_{i\oplus 1}^{1} = \widetilde{c}_{i\oplus 1}^{1} \tag{13.11}$$
$$c_{i\oplus 1}^{2} = \widetilde{c}_{i\oplus 1}^{2} \tag{13.12}$$
$$c_{i\oplus 1}^{3} = \widetilde{c}_{i\oplus 1}^{3} \tag{13.13}$$
$$c_{i\oplus 1}^{4} = \widetilde{c}_{i\oplus 1}^{4} \tag{13.14}$$
$$c_{i\oplus 1}^{5} = \widetilde{c}_{i\oplus 1}^{5} \oplus \widetilde{c}_{i\oplus 1}^{6} \oplus \widetilde{c}_{i\oplus 1}^{7} \oplus \widetilde{c}_{i\oplus 1}^{8} \oplus \widetilde{c}_{i\oplus 1}^{9} \oplus \widetilde{c}_{i\oplus 1}^{10} \tag{13.15}$$

13.2 Second-Order KSA

13.2.1 AND (1. Step)

$$\widetilde{g}_i^1 = a_i^2 b_i^2 \oplus a_i^1 b_i^2 \oplus a_i^2 b_i^1 \oplus m_i^1 \tag{13.16}$$

$$\widetilde{g}_i^2 = a_i^3 b_i^3 \oplus a_i^1 b_i^3 \oplus a_i^3 b_i^1 \oplus m_i^2 \tag{13.17}$$

$$\widetilde{g}_i^3 = a_i^4 b_i^4 \oplus a_i^1 b_i^4 \oplus a_i^4 b_i^1 \oplus m_i^3 \tag{13.18}$$

$$\widetilde{g}_i^4 = a_i^1 b_i^1 \oplus a_i^1 b_i^5 \oplus a_i^5 b_i^1 \oplus m_i^4 \tag{13.19}$$

$$\widetilde{g}_i^5 = a_i^2 b_i^3 \oplus a_i^3 b_i^2 \tag{13.20}$$

$$\widetilde{g}_i^6 = a_i^2 b_i^4 \oplus a_i^4 b_i^2 \oplus m_i^1 \tag{13.21}$$

$$\widetilde{g}_i^7 = a_i^5 b_i^5 \oplus a_i^2 b_i^5 \oplus a_i^5 b_i^2 \tag{13.22}$$

$$\widetilde{g}_i^8 = a_i^3 b_i^4 \oplus a_i^4 b_i^3 \oplus m_i^2 \tag{13.23}$$

$$\widetilde{g}_i^9 = a_i^3 b_i^5 \oplus a_i^5 b_i^3 \oplus m_i^3 \tag{13.24}$$

$$\widetilde{g}_i^{10} = a_i^4 b_i^5 \oplus a_i^5 b_i^4 \oplus m_i^4 \tag{13.25}$$

13.2.2 AND/XOR (1. Step)

$$\widetilde{g}_{i:j}^1 = g_i^2 \oplus g_j^2 p_i^2 \oplus g_j^1 p_i^2 \oplus g_j^2 p_i^1 \tag{13.26}$$

$$\widetilde{g}_{i:j}^2 = g_i^3 \oplus g_j^3 p_i^3 \oplus g_j^1 p_i^3 \oplus g_j^3 p_i^1 \tag{13.27}$$

$$\widetilde{g}_{i:j}^3 = g_i^4 \oplus g_j^4 p_i^4 \oplus g_j^1 p_i^4 \oplus g_j^4 p_i^1 \tag{13.28}$$

$$\widetilde{g}_{i:j}^4 = g_i^1 \oplus g_j^1 p_i^1 \oplus g_j^1 p_i^5 \oplus g_j^5 p_i^1 \tag{13.29}$$

$$\widetilde{g}_{i:j}^5 = g_j^2 p_i^3 \oplus g_j^3 p_i^2 \tag{13.30}$$

$$\widetilde{g}_{i:j}^6 = g_j^2 p_i^4 \oplus g_j^4 p_i^2 \tag{13.31}$$

$$\widetilde{g}_{i:j}^7 = g_i^5 \oplus g_j^5 p_i^5 \oplus g_j^2 p_i^5 \oplus g_j^5 p_i^2 \tag{13.32}$$

$$\widetilde{g}_{i:j}^8 = g_j^3 p_i^4 \oplus g_j^4 p_i^3 \tag{13.33}$$

$$\widetilde{g}_{i:j}^9 = g_j^3 p_i^5 \oplus g_j^5 p_i^3 \tag{13.34}$$

$$\widetilde{g}_{i:j}^{10} = g_j^4 p_i^5 \oplus g_j^5 p_i^4 \tag{13.35}$$

Chapter 14

Specifications of the Selected Sboxes

14.1 Algebraic Degree

Table 14.1: Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_1$.

Order	1	2	3	4	5	6	7
Count	0	0	3	4	120	128	0

Table 14.2: Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_2$.

Order	1	2	3	4	5	6	7
Count	0	7	24	224	0	0	0

Table 14.3: Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_3$.

Order	1	2	3	4	5	6	7
Count	0	0	0	0	0	63	192

Table 14.4: Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_4$.

Order	1	2	3	4	5	6	7
Count	0	0	0	0	0	15	240

Table 14.5: Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_5$.

Order	1	2	3	4	5	6	7
Count	0	0	0	0	0	3	252

Table 14.6: Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_6$.

Order	1	2	3	4	5	6	7
Count	0	0	0	0	1	62	192

14.2 Look-up tables

Table 14.7: The round function of $\mathbf{SB}_1(xy)$. 8 iterations of it result in the final Sbox.

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	80	C0	84	C4	00	40	04	44	A0	E0	E4	A4	24	64	60	20
	1	81	C1	85	C5	01	41	05	45	A1	E1	E5	A5	25	65	61	21
	2	90	D0	94	D4	10	50	14	54	B0	F0	F4	B4	34	74	70	30
	3	91	D1	95	D5	11	51	15	55	B1	F1	F5	B5	35	75	71	31
	4	82	C2	86	C6	02	42	06	46	A2	E2	E6	A6	26	66	62	22
	5	83	C3	87	C7	03	43	07	47	A3	E3	E7	A7	27	67	63	23
	6	92	D2	96	D6	12	52	16	56	B2	F2	F6	B6	36	76	72	32
	7	93	D3	97	D7	13	53	17	57	B3	F3	F7	B7	37	77	73	33
	8	88	C8	8C	CC	08	48	0C	4C	A8	E8	EC	AC	2C	6C	68	28
	9	89	C9	8D	CD	09	49	0D	4D	A9	E9	ED	AD	2D	6D	69	29
	A	98	D8	9C	DC	18	58	1C	5C	B8	F8	FC	BC	3C	7C	78	38
	B	99	D9	9D	DD	19	59	1D	5D	B9	F9	FD	BD	3D	7D	79	39
	C	8A	CA	8E	CE	0A	4A	0E	4E	AA	EA	EE	AE	2E	6E	6A	2A
	D	8B	CB	8F	CF	0B	4B	0F	4F	AB	EB	EF	AF	2F	6F	6B	2B
	E	9A	DA	9E	DE	1A	5A	1E	5E	BA	FA	FE	BE	3E	7E	7A	3A
	F	9B	DB	9F	DF	1B	5B	1F	5F	BB	FB	FF	BF	3F	7F	7B	3B

Table 14.8: The round function of $\mathbf{SB}_2(xy)$. 2 iterations of it result in the final Sbox.

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	22	60	A6	E4	1A	58	DC	9E	41	03	79	3B	FD	BF	87	C5
	1	06	44	82	C0	3E	7C	F8	BA	65	27	5D	1F	D9	9B	A3	E1
	2	6A	28	EE	AC	52	10	94	D6	09	4B	31	73	B5	F7	CF	8D
	3	4E	0C	CA	88	76	34	B0	F2	2D	6F	15	57	91	D3	EB	A9
	4	A1	E3	25	67	99	DB	5F	1D	C2	80	FA	B8	7E	3C	04	46
	5	85	C7	01	43	BD	FF	7B	39	E6	A4	DE	9C	5A	18	20	62
	6	CD	8F	49	0B	F5	B7	33	71	AE	EC	96	D4	12	50	68	2A
	7	E9	AB	6D	2F	D1	93	17	55	8A	C8	B2	F0	36	74	4C	0E
	8	14	56	90	D2	2C	6E	EA	A8	77	35	4F	0D	CB	89	B1	F3
	9	30	72	B4	F6	08	4A	CE	8C	53	11	6B	29	EF	AD	95	D7
	A	97	D5	13	51	AF	ED	69	2B	F4	B6	CC	8E	48	0A	32	70
	B	B3	F1	37	75	8B	C9	4D	0F	D0	92	E8	AA	6C	2E	16	54
	C	DF	9D	5B	19	E7	A5	21	63	BC	FE	84	C6	00	42	7A	38
	D	FB	B9	7F	3D	C3	81	05	47	98	DA	A0	E2	24	66	5E	1C
	E	78	3A	FC	BE	40	02	86	C4	1B	59	23	61	A7	E5	DD	9F
	F	5C	1E	D8	9A	64	26	A2	E0	3F	7D	07	45	83	C1	F9	BB

Table 14.9: The round function of $\mathbf{SB}_3(xy)$. 4 iterations of it result in the final Sbox.

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	77	5C	32	19	FD	D6	B8	93	40	05	CA	8F	2E	6B	A4	E1
	1	54	7F	11	3A	DE	F5	9B	B0	63	26	E9	AC	0D	48	87	C2
	2	31	1A	74	5F	BB	90	FE	D5	06	43	8C	C9	68	2D	E2	A7
	3	12	39	57	7C	98	B3	DD	F6	25	60	AF	EA	4B	0E	C1	84
	4	FB	D0	BE	95	71	5A	34	1F	CC	89	46	03	A2	E7	28	6D
	5	D8	F3	9D	B6	52	79	17	3C	EF	AA	65	20	81	C4	0B	4E
	6	9E	B5	DB	F0	14	3F	51	7A	A9	EC	23	66	C7	82	4D	08
	7	BD	96	F8	D3	37	1C	72	59	8A	CF	00	45	E4	A1	6E	2B
	8	4C	67	09	22	C6	ED	83	A8	7B	3E	F1	B4	15	50	9F	DA
	9	6F	44	2A	01	E5	CE	A0	8B	58	1D	D2	97	36	73	BC	F9
	A	C0	EB	85	AE	4A	61	0F	24	F7	B2	7D	38	99	DC	13	56
	B	E3	C8	A6	8D	69	42	2C	07	D4	91	5E	1B	BA	FF	30	75
	C	86	AD	C3	E8	0C	27	49	62	B1	F4	3B	7E	DF	9A	55	10
	D	A5	8E	E0	CB	2F	04	6A	41	92	D7	18	5D	FC	B9	76	33
	E	29	02	6C	47	A3	88	E6	CD	1E	5B	94	D1	70	35	FA	BF
	F	0A	21	4F	64	80	AB	C5	EE	3D	78	B7	F2	53	16	D9	9C

Table 14.10: The round function of $\mathbf{SB}_4(xy)$. 5 iterations of it result in the final Sbox.

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	60	70	40	50	20	30	00	10	E0	F0	C0	D0	A0	B0	80	90
	1	21	31	01	11	61	71	41	51	A1	B1	81	91	E1	F1	C1	D1
	2	72	62	52	42	32	22	12	02	F2	E2	D2	C2	B2	A2	92	82
	3	13	03	33	23	53	43	73	63	93	83	B3	A3	D3	C3	F3	E3
	4	64	74	44	54	24	34	04	14	E4	F4	C4	D4	A4	B4	84	94
	5	A5	B5	85	95	E5	F5	C5	D5	25	35	05	15	65	75	45	55
	6	E6	F6	C6	D6	A6	B6	86	96	66	76	46	56	26	36	06	16
	7	07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7
	8	58	48	78	68	18	08	38	28	D8	C8	F8	E8	98	88	B8	A8
	9	89	99	A9	B9	C9	D9	E9	F9	09	19	29	39	49	59	69	79
	A	5A	4A	7A	6A	1A	0A	3A	2A	DA	CA	FA	EA	9A	8A	BA	AA
	B	AB	BB	8B	9B	EB	FB	CB	DB	2B	3B	0B	1B	6B	7B	4B	5B
	C	1C	0C	3C	2C	5C	4C	7C	6C	9C	8C	BC	AC	DC	CC	FC	EC
	D	4D	5D	6D	7D	0D	1D	2D	3D	CD	DD	ED	FD	8D	9D	AD	BD
	E	8E	9E	AE	BE	CE	DE	EE	FE	0E	1E	2E	3E	4E	5E	6E	7E
	F	FF	EF	DF	CF	BF	AF	9F	8F	7F	6F	5F	4F	3F	2F	1F	0F

Table 14.11: The round function of $\mathbf{SB}_5(xy)$. 9 iterations of it result in the final Sbox.

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	64	60	65	61	24	20	25	21	74	70	71	75	35	31	30	34
	1	E4	E0	E5	E1	A4	A0	A5	A1	F4	F0	F1	F5	B5	B1	B0	B4
	2	66	62	67	63	26	22	27	23	76	72	73	77	37	33	32	36
	3	E6	E2	E7	E3	A6	A2	A7	A3	F6	F2	F3	F7	B7	B3	B2	B6
	4	6C	68	6D	69	2C	28	2D	29	7C	78	79	7D	3D	39	38	3C
	5	EC	E8	ED	E9	AC	A8	AD	A9	FC	F8	F9	FD	BD	B9	B8	BC
	6	6E	6A	6F	6B	2E	2A	2F	2B	7E	7A	7B	7F	3F	3B	3A	3E
	7	EE	EA	EF	EB	AE	AA	AF	AB	FE	FA	FB	FF	BF	BB	BA	BE
	8	44	40	45	41	04	00	05	01	54	50	51	55	15	11	10	14
	9	C4	C0	C5	C1	84	80	85	81	D4	D0	D1	D5	95	91	90	94
	A	46	42	47	43	06	02	07	03	56	52	53	57	17	13	12	16
	B	C6	C2	C7	C3	86	82	87	83	D6	D2	D3	D7	97	93	92	96
	C	CC	C8	CD	C9	8C	88	8D	89	DC	D8	D9	DD	9D	99	98	9C
	D	4C	48	4D	49	0C	08	0D	09	5C	58	59	5D	1D	19	18	1C
	E	CE	CA	CF	CB	8E	8A	8F	8B	DE	DA	DB	DF	9F	9B	9A	9E
	F	4E	4A	4F	4B	0E	0A	0F	0B	5E	5A	5B	5F	1F	1B	1A	1E

Table 14.12: The round function of $\mathbf{SB}_6(xy)$. 4 iterations of it result in the final Sbox.

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	E6	B9	4B	14	9F	C0	32	6D	07	58	F5	AA	D3	8C	21	7E
	1	EB	B4	46	19	92	CD	3F	60	0A	55	F8	A7	DE	81	2C	73
	2	EF	B0	42	1D	96	C9	3B	64	0E	51	FC	A3	DA	85	28	77
	3	E2	BD	4F	10	9B	C4	36	69	03	5C	F1	AE	D7	88	25	7A
	4	E7	B8	4A	15	9E	C1	33	6C	06	59	F4	AB	D2	8D	20	7F
	5	EA	B5	47	18	93	CC	3E	61	0B	54	F9	A6	DF	80	2D	72
	6	E3	BC	4E	11	9A	C5	37	68	02	5D	F0	AF	D6	89	24	7B
	7	EE	B1	43	1C	97	C8	3A	65	0F	50	FD	A2	DB	84	29	76
	8	E4	BB	49	16	9D	C2	30	6F	05	5A	F7	A8	D1	8E	23	7C
	9	E9	B6	44	1B	90	CF	3D	62	08	57	FA	A5	DC	83	2E	71
	A	E5	BA	48	17	9C	C3	31	6E	04	5B	F6	A9	D0	8F	22	7D
	B	E8	B7	45	1A	91	CE	3C	63	09	56	FB	A4	DD	82	2F	70
	C	EC	B3	41	1E	95	CA	38	67	0D	52	FF	A0	D9	86	2B	74
	D	E1	BE	4C	13	98	C7	35	6A	00	5F	F2	AD	D4	8B	26	79
	E	E0	BF	4D	12	99	C6	34	6B	01	5E	F3	AC	D5	8A	27	78
	F	ED	B2	40	1F	94	CB	39	66	0C	53	FE	A1	D8	87	2A	75

Chapter 15

Shared Functions of the Sbox

15.1 Shared Functions Sbox

In this section we give the shared representation of the functions used for the Sbox with three shares.

$$\begin{aligned}
&G_1(d_2, c_2, b_2, a_2, d_3, c_3, b_3, a_3) = (h_1, g_1, f_1, e_1) \\
e_1 &= a_2 + b_2 + c_2 + d_2 + b_3 + c_2b_3 + b_2c_3 + b_3c_3 \\
f_1 &= a_2 \\
g_1 &= 1 + a_2 + d_2 + b_3 + c_2b_3 + b_2c_3 + b_3c_3 \\
h_1 &= 1 + a_2 + b_2 + b_3 + c_2b_3 + d_2b_3 + b_2c_3 + d_2c_3 + b_3c_3 \\
&\quad + b_2d_3 + c_2d_3 + b_3d_3 + c_3d_3
\end{aligned} \tag{15.1}$$

$$\begin{aligned}
&G_2(d_3, c_3, b_3, a_3, d_1, c_1, b_1, a_1) = (h_2, g_2, f_2, e_2) \\
e_2 &= a_3 + b_3 + c_3 + d_3 + b_1 + c_3b_1 + b_3c_1 + b_1c_1 \\
f_2 &= a_3 \\
g_2 &= 1 + a_3 + d_3 + b_1 + c_3b_1 + b_3c_1 + b_1c_1 \\
h_2 &= 1 + a_3 + b_3 + b_1 + c_3b_1 + d_3b_1 + b_3c_1 + d_3c_1 + b_1c_1 \\
&\quad + b_3d_1 + c_3d_1 + b_1d_1 + c_1d_1
\end{aligned} \tag{15.2}$$

$$\begin{aligned}
&G_3(d_1, c_1, b_1, a_1, d_2, c_2, b_2, a_2) = (h_3, g_3, f_3, e_3) \\
e_3 &= a_1 + b_1 + c_1 + d_1 + b_2 + c_1b_2 + b_1c_2 + b_2c_2 \\
f_3 &= a_1 \\
g_3 &= 1 + a_1 + d_1 + b_2 + c_1b_2 + b_1c_2 + b_2c_2 \\
h_3 &= 1 + a_1 + b_1 + b_2 + c_1b_2 + d_1b_2 + b_1c_2 + d_1c_2 + b_2c_2 \\
&\quad + b_1d_2 + c_1d_2 + b_2d_2 + c_2d_2
\end{aligned} \tag{15.3}$$

$$\begin{aligned}
&F_1(d_2, c_2, b_2, a_2, d_3, c_3, b_3, a_3) = (h_1, g_1, f_1, e_1) \\
e_1 &= a_2 \\
f_1 &= c_2 + d_2 + d_2b_3 + b_2d_3 + b_3d_3 \\
g_1 &= 1 + a_2 + b_2 + c_2 + d_2c_3 + c_2d_3 + c_3d_3 \\
h_1 &= c_2 + d_2b_3 + b_2d_3 + b_3d_3
\end{aligned} \tag{15.4}$$

$$\begin{aligned}
&F_2(d_3, c_3, b_3, a_3, d_1, c_1, b_1, a_1) = (h_2, g_2, f_2, e_2) \\
e_2 &= a_3 \\
f_2 &= c_3 + d_3 + d_3b_1 + b_3d_1 + b_1d_1 \\
g_2 &= 1 + a_3 + b_3 + c_3 + d_3c_1 + c_3d_1 + c_1d_1 \\
h_2 &= c_3 + d_3b_1 + b_3d_1 + b_1d_1
\end{aligned} \tag{15.5}$$

$$\begin{aligned}
&F_3(d_1, c_1, b_1, a_1, d_2, c_2, b_2, a_2) = (h_3, g_3, f_3, e_3) \\
e_3 &= a_1 \\
f_3 &= c_1 + d_1 + d_1b_2 + b_1d_2 + b_2d_2 \\
g_3 &= 1 + a_1 + b_1 + c_1 + d_1c_2 + c_1d_2 + c_2d_2 \\
h_3 &= c_1 + d_1b_2 + b_1d_2 + b_2d_2
\end{aligned} \tag{15.6}$$

$$\begin{aligned}
&R_1(d_2, c_2, b_2, a_2, d_3, c_3, b_3, a_3) = (h_1, g_1, f_1, e_1) \\
e_1 &= a_2 + b_2 + d_2b_3 + d_2c_3 + b_2d_3 + c_2d_3 + b_3d_3 + c_3d_3 \\
f_1 &= b_2 + c_2 \\
g_1 &= c_2 + b_2a_3 + c_2a_3 + a_2b_3 + a_3b_3 + a_2c_3 + a_3c_3 \\
h_1 &= d_2 + b_3 + c_2b_3 + b_2c_3 + b_3c_3
\end{aligned} \tag{15.7}$$

$$\begin{aligned}
&R_2(d_3, c_3, b_3, a_3, d_1, c_1, b_1, a_1) = (h_2, g_2, f_2, e_2) \\
e_2 &= a_3 + b_3 + d_3b_1 + d_3c_1 + b_3d_1 + c_3d_1 + b_1d_1 + c_1d_1 \\
f_2 &= b_3 + c_3 \\
g_2 &= c_3 + b_3a_1 + c_3a_1 + a_3b_1 + a_1b_1 + a_3c_1 + a_1c_1 \\
h_2 &= d_3 + b_1 + c_3b_1 + b_3c_1 + b_1c_1
\end{aligned} \tag{15.8}$$

$$\begin{aligned}
R_3(d_1, c_1, b_1, a_1, d_2, c_2, b_2, a_2) &= (h_3, g_3, f_3, e_3) \\
e_3 &= a_1 + b_1 + d_1b_2 + d_1c_2 + b_1d_2 + c_1d_2 + b_2d_2 + c_2d_2 \\
f_3 &= b_1 + c_1 \\
g_3 &= c_1 + b_1a_2 + c_1a_2 + a_1b_2 + a_2b_2 + a_1c_2 + a_2c_2 \\
h_3 &= d_1 + b_2 + c_1b_2 + b_1c_2 + b_2c_2
\end{aligned} \tag{15.9}$$

$$\begin{aligned}
&Q_1(d_2, c_2, b_2, a_2, d_3, c_3, b_3, a_3) = (h_1, g_1, f_1, e_1) \\
e_1 &= 1 + a_2 + b_3 + c_3 + d_2b_3 + d_2c_3 + b_2d_3 + c_2d_3 + b_3d_3 + c_3d_3 \\
f_1 &= 1 + a_2 + b_2 \\
g_1 &= a_2 + b_2 + c_2 + d_2 + b_3 + c_3 + d_2b_3 + d_2c_3 + b_2d_3 + c_2d_3 + b_3d_3 + c_3d_3 \\
h_1 &= d_2 + c_3 + d_3 + b_2a_3 + c_2a_3 + d_2a_3 + a_2b_3 + c_2b_3 + d_2b_3 + a_3b_3 \\
&\quad + a_2c_3 + b_2c_3 + a_3c_3 + b_3c_3 + a_2d_3 + b_2d_3 + a_3d_3 + b_3d_3
\end{aligned} \tag{15.10}$$

$$\begin{aligned}
&Q_2(d_3, c_3, b_3, a_3, d_1, c_1, b_1, a_1) = (h_2, g_2, f_2, e_2) \\
e_2 &= 1 + a_3 + b_1 + c_1 + d_3b_1 + d_3c_1 + b_3d_1 + c_3d_1 + b_1d_1 + c_1d_1 \\
f_2 &= 1 + a_3 + b_3 \\
g_2 &= a_3 + b_3 + c_3 + d_3 + b_1 + c_1 + d_3b_1 + d_3c_1 + b_3d_1 + c_3d_1 + b_1d_1 + c_1d_1 \\
h_2 &= d_3 + c_1 + d_1 + b_3a_1 + c_3a_1 + d_3a_1 + a_3b_1 + c_3b_1 + d_3b_1 + a_1b_1 \\
&\quad + a_3c_1 + b_3c_1 + a_1c_1 + b_1c_1 + a_3d_1 + b_3d_1 + a_1d_1 + b_1d_1
\end{aligned} \tag{15.11}$$

$$\begin{aligned}
&Q_3(d_1, c_1, b_1, a_1, d_2, c_2, b_2, a_2) = (h_3, g_3, f_3, e_3) \\
e_3 &= 1 + a_1 + b_2 + c_2 + d_1b_2 + d_1c_2 + b_1d_2 + c_1d_2 + b_2d_2 + c_2d_2 \\
f_3 &= 1 + a_1 + b_1 \\
g_3 &= a_1 + b_1 + c_1 + d_1 + b_2 + c_2 + d_1b_2 + d_1c_2 + b_1d_2 + c_1d_2 + b_2d_2 + c_2d_2 \\
h_3 &= d_1 + c_2 + d_2 + b_1a_2 + c_1a_2 + d_1a_2 + a_1b_2 + c_1b_2 + d_1b_2 + a_2b_2 \\
&\quad + a_1c_2 + b_1c_2 + a_2c_2 + b_2c_2 + a_1d_2 + b_1d_2 + a_2d_2 + b_2d_2
\end{aligned} \tag{15.12}$$

Bibliography

[ABF13] Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. Leakage-Resilient Symmetric Encryption via Re-keying. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 471–488. Springer, 2013. 11

[AHMP08] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C-W Phan. SHA-3 proposal BLAKE. *Submission to NIST*, 2008. 92

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010. 3

[BB02] Elad Barkan and Eli Biham. In How Many Ways Can You Write Rijndael? In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 160–175. Springer, 2002. 112

[BB16] Paul Bottinelli and Joppe W. Bos. Computational aspects of correlation power analysis. *Journal of Cryptographic Engineering*, pages 1–15, 2016. 46, 47, 49, 51, 60

[BBD$^+$14] Shivam Bhasin, Nicolas Bruneau, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. Analysis and Improvements of the DPA Contest v4 Implementation. In Rajat Subhra Chakraborty, Vashek Matyas, and Patrick Schaumont, editors, *Security, Privacy, and Applied Cryptography Engineering - 4th International Conference, SPACE 2014, Pune, India, October 18-22, 2014. Proceedings*, volume 8804 of *Lecture Notes in Computer Science*, pages 201–218. Springer, 2014. 40

[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. 157

[BBI$^+$15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A Block Cipher for Low Energy. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology*

- ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015. 116

[BBK+03] Guido Bertoni, Luca Breveglieri, Israel Koren, Paolo Maistri, and Vincenzo Piuri. Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. *IEEE Trans. Computers*, 52(4):492–505, 2003. 127

[BBK+13] Begül Bilgin, Andrey Bogdanov, Miroslav Knezevic, Florian Mendel, and Qingju Wang. Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 142–158. Springer, 2013. 158

[BCC+14] Julien Bringer, Claude Carlet, Hervé Chabanne, Sylvain Guilley, and Houssem Maghrebi. Orthogonal Direct Sum Masking - A Smartcard Friendly Computation Paradigm in a Code, with Builtin Protection against Side-Channel and Fault Attacks. In David Naccache and Damien Sauveron, editors, *Information Security Theory and Practice. Securing the Internet of Things - 8th IFIP WG 11.2 International Workshop, WISTP 2014, Heraklion, Crete, Greece, June 30 - July 2, 2014. Proceedings*, volume 8501 of *Lecture Notes in Computer Science*, pages 40–56. Springer, 2014. 9, 128, 156

[BCL12] Julien Bringer, Hervé Chabanne, and Thanh-Ha Le. Protecting AES against side-channel analysis using wire-tap codes. *J. Cryptographic Engineering*, 2(2):129–141, 2012. 128

[BCO04] 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 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004. 6, 13, 25, 45, 65

[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 Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 187–199. Springer, 2013. 87

[Ber08a] Daniel J Bernstein. ChaCha, a variant of Salsa20. In *Workshop Record of SASC*, volume 8, 2008. 92, 104, 156

[Ber08b] Daniel J. Bernstein. The Salsa20 Family of Stream Ciphers. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Fi-*

nalists, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2008. 92

[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. 6, 12, 26, 157

[BGG+16a] Erik Boss, Vincent Grosso, Tim Güneysu, Gregor Leander, Amir Moradi, and Tobias Schneider. Strong 8-bit Sboxes with Efficient Masking in Hardware. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 171–193. Springer, 2016. 5, 8, 87, 105, 156

[BGG+16b] Erik Boss, Vincent Grosso, Tim Güneysu, Gregor Leander, Amir Moradi, and Tobias Schneider. Strong 8-bit Sboxes with Efficient Masking in Hardware. Cryptology ePrint Archive, Report 2016/647, 2016. http://eprint.iacr.org/2016/647.

[BGN+14a] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. A More Efficient AES Threshold Implementation. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, volume 8469 of *Lecture Notes in Computer Science*, pages 267–284. Springer, 2014. 64, 87, 88, 106, 134, 142, 145

[BGN+14b] 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 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014. 6, 26, 27, 41, 87, 96, 98, 100, 102, 113

[BGN+15] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Trade-Offs for Threshold Implementations Illustrated on AES. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(7):1188–1200, 2015. 106, 110, 111, 117

[BGP+11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual Information Analysis: a Comprehensive Study. *J. Cryptology*, 24(2):269–291, 2011. 66, 67

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY

Family of Block Ciphers and Its Low-Latency Variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016. 110, 116

[BKL$^+$07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007. 8, 73, 74, 142

[Bla03] Richard E Blahut. *Algebraic codes for data transmission.* Cambridge Univ. Press, Cambridge, 2003. 130

[BLV12] Christina Boura, Sylvain Lévêque, and David Vigilant. Side-Channel Analysis of Grøstl and Skein. In *2012 IEEE Symposium on Security and Privacy Workshops, San Francisco, CA, USA, May 24-25, 2012*, pages 16–26. IEEE Computer Society, 2012. 92

[BNN$^+$12] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, and Georg Stütz. Threshold Implementations of All 3 ×3 and 4 ×4 S-Boxes. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2012. 95, 97, 99, 106, 133, 142, 148

[BNN$^+$15] 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. 8, 87, 88, 109, 110, 111, 112, 113, 117, 119, 133, 135, 148

[Bog08] Andrey Bogdanov. Multiple-Differential Side-Channel Collision Attacks on AES. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 30–44. Springer, 2008. 54

[BP10a] Olivier Benoît and Thomas Peyrin. Side-Channel Analysis of Six SHA-3 Candidates. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 140–157. Springer, 2010. 92

[BP10b] Joan Boyar and René Peralta. A New Combinational Logic Minimization Technique with Applications to Cryptology. In Paola Festa, editor, *Experimental Algorithms, 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy,* *May 20-22, 2010. Proceedings*, volume 6049 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 2010. 117

[BR00] Paulo S. L. M. Barreto and Vincent Rijmen. The Khazad legacy-level block cipher. *Primitive submitted to NESSIE*, 97, 2000. 106, 117

[Bra] Pierce Brady. pearspdf. http://www.mathworks.com/matlabcentral/fileexchange/26516-pearspdf. 72

[Bri08] Marcus Brinkmann. EA classification of all 4 bit functions. personal communication, 2008. 118

[BS90] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990. 108

[BS97] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997. 4, 127

[BSMG17] Florian Bache, Tobias Schneider, Amir Moradi, and Tim Güneysu. 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. 157

[Can05] David Canright. A Very Compact S-Box for AES. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005. 111

[CBR$^+$15] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, Ventzislav Nikov, and Svetla Nikova. Higher-Order Threshold Implementation of the AES S-Box. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 259–272. Springer, 2015. 87, 88

[CCG10] Konstantinos Chatzikokolakis, Tom Chothia, and Apratim Guha. Statistical Measurement of Information Leakage. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March*

20-28, 2010. Proceedings, volume 6015 of *Lecture Notes in Computer Science*, pages 390–404. Springer, 2010. 26

[CDG+13] Jeremy Cooper, Elke Demulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, and Pankaj Rohatgi. Test Vector Leakage Assessment (TVLA) Methodology in Practice. International Cryptographic Module Conference, 2013. 30, 38, 155

[CDL15] Anne Canteaut, Sébastien Duval, and Gaëtan Leurent. Construction of Lightweight S-Boxes Using Feistel and MISTY Structures. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566 of *Lecture Notes in Computer Science*, pages 373–393. Springer, 2015. 106, 109, 115, 118

[CFE16] Cong Chen, Mohammad Farmani, and Thomas Eisenbarth. A tale of two shares: Why two-share threshold implementation seems worthwhile-and why it is not. Cryptology ePrint Archive, Report 2016/434, 2016. `https://eprint.iacr.org/2016/434`. 89, 158

[CFGR10] Christophe Clavier, Benoit Feix, Georges Gagnerot, and Mylène Roussellet. Passive and Active Combined Attacks on AES Combining Fault Attacks and Side Channel Analysis. In Luca Breveglieri, Marc Joye, Israel Koren, David Naccache, and Ingrid Verbauwhede, editors, *2010 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2010, Santa Barbara, California, USA, 21 August 2010*, pages 10–19. IEEE Computer Society, 2010. 128

[CG11] Tom Chothia and Apratim Guha. A Statistical Test for Information Leaks Using Continuous Mutual Information. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011*, pages 177–190. IEEE Computer Society, 2011. 26

[CGP+12] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of Security Proofs from One Leakage Model to Another: A New Issue. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, volume 7275 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2012. 64

[CGTV15] Jean-Sébastien Coron, Johann Großschädl, Mehdi Tibouchi, and Praveen Kumar Vadnala. Conversion from Arithmetic to Boolean Masking with Logarithmic Complexity. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 130–149. Springer, 2015. 7, 92, 96, 97, 99, 104

[CGV14] Jean-Sébastien Coron, Johann Großschädl, and Praveen Kumar Vadnala. Secure Conversion between Boolean and Arithmetic Masking of Any Order. In

Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 188–205. Springer, 2014. 92, 93, 96, 97, 99

[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, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999. 4, 11, 12

[CRB$^+$16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with d+1 Shares in Hardware. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 194–212. Springer, 2016. 89, 158

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002. 7, 13, 64

[CTO$^+$14] Mathieu Carbone, Sébastien Tiran, Sébastien Ordas, Michel Agoyan, Yannick Teglia, Gilles R. Ducharme, and Philippe Maurine. On Adaptive Bandwidth Selection for Efficient MIA. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*, pages 82–97. Springer, 2014. 67

[CV94] Florent Chabaud and Serge Vaudenay. Links Between Differential and Linear Cryptanalysis. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 356–365. Springer, 1994. 107

[DDP13] Guillaume Dabosville, Julien Doget, and Emmanuel Prouff. A New Second-Order Side Channel Attack Based on Linear Regression. *IEEE Trans. Computers*, 62(8):1629–1640, 2013. 66

[Deb12] Blandine Debraize. Efficient and Provably Secure Methods for Switching from Arithmetic to Boolean Masking. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2012. 92

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015. 54, 64, 80, 84

[DPAR00] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. Nessie proposal: NOEKEON. In *First Open NESSIE Workshop*, pages 213–230, 2000. 106

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptographic Engineering*, 1(2):123–144, 2011. 66

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. 117

[DS16] François Durvaux and François-Xavier Standaert. From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 240–262. Springer, 2016. 15, 64

[DSV14] François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. How to Certify the Leakage of a Chip? In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 459–476. Springer, 2014. 64, 68, 76, 77

[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 Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 34–50. Springer, 2015. 43, 54

[DV12] François Dassance and Alexandre Venelli. Combined Fault and Side-Channel Attacks on the AES Key Schedule. In Guido Bertoni and Benedikt Gierlichs, editors, *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, Leuven, Belgium, September 9, 2012*, pages 63–71. IEEE Computer Society, 2012. 128, 135

[EKM$^+$08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoqCode Hopping Scheme. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 203–220. Springer, 2008. 4

[FD81] David Freedman and Persi Diaconis. On the histogram as a density estimator:L2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 57(4):453–476, 1981. 67

[FGP+17] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults and the robust probing model. Cryptology ePrint Archive, Report 2017/711, 2017. `https://eprint.iacr.org/2017/711`. 157

[FLS+10] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein Hash Function Family. *http://www.skein-hash.info/sites/default/files/skein1.3.pdf*, 2010. 7, 92, 156

[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. *Proceedings of SASC*, pages 123–150, 2008. 92

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual Information Analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008. 7, 25, 66

[GGNS13] Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert. Block Ciphers That Are Easier to Mask: How Far Can We Go? In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2013. 106

[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. 5, 26, 31, 43, 99, 155

[GLS+] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici, Anthony Journault, François Durvaux, Lubos Gaspar, and Stéphanie Kerckhof. SCREAM Side-Channel Resistant Authenticated Encryption with Masking – ver 3. submission to CAESAR competition of authenticated ciphers, `https://competitions.cr.yp.to/round2/screamv3.pdf`. 117

[GLSV14] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations.

In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 18–37. Springer, 2014. 106, 117

[GM17] Hannes Groß and Stefan Mangard. Reconciling d+1 masking in hardware and software. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 115–136. Springer, 2017. 158

[GMJK15] Xiaofei Guo, Debdeep Mukhopadhyay, Chenglu Jin, and Ramesh Karri. Security analysis of concurrent error detection against differential fault analysis. *J. Cryptographic Engineering*, 5(3):153–169, 2015. 130, 140

[GMK16] 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. https://eprint.iacr.org/2016/486. 89, 158

[Gol07] Jovan Dj. Golic. Techniques for Random Masking in Hardware. *IEEE Trans. on Circuits and Systems*, 54-I(2):291–300, 2007. 8, 92

[Gou01] Louis Goubin. A Sound Method for Switching between Boolean and Arithmetic Masking. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 3–15. Springer, 2001. 7, 92, 156

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011. 9, 120, 129, 143, 156

[Gru72] Eli Grushka. Characterization of exponentially modified Gaussian peaks in chromatography. *Analytical Chemistry*, 44(11):1733–1738, 1972. PMID: 22324584. 68

[GST12] Benedikt Gierlichs, Jörn-Marc Schmidt, and Michael Tunstall. Infective Computation and Dummy Rounds: Fault Protection for Block Ciphers without Check-before-Output. In Alejandro Hevia and Gregory Neven, editors, *Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings*, volume 7533 of *Lecture Notes in Computer Science*, pages 305–321. Springer, 2012. 128

[Hig02] Nicholas J. Higham. *Accuracy and stability of numerical algorithms (2. ed.)*. SIAM, 2002. 31, 47

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003. 11, 12, 123, 157

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. 11, 25, 29

[KKG03] Ramesh Karri, Grigori Kuznetsov, and Michael Gössel. Parity-Based Concurrent Error Detection of Substitution-Permutation Network Block Ciphers. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 113–124. Springer, 2003. 127

[KKT04a] Mark G. Karpovsky, Konrad J. Kulikowski, and Alexander Taubin. Differential Fault Analysis Attack Resistant Architectures for the Advanced Encryption Standard. In Jean-Jacques Quisquater, Pierre Paradinas, Yves Deswarte, and Anas Abou El Kalam, editors, *Smart Card Research and Advanced Applications VI, IFIP 18th World Computer Congress, TC8/WG8.8 & TC11/WG11.2 Sixth International Conference on Smart Card Research and Advanced Applications (CARDIS), 22-27 August 2004, Toulouse, France*, volume 153 of *IFIP*, pages 177–192. Kluwer/Springer, 2004. 127

[KKT04b] Mark G. Karpovsky, Konrad J. Kulikowski, and Alexander Taubin. Robust Protection against Fault-Injection Attacks on Smart Cards Implementing the Advanced Encryption Standard. In *2004 International Conference on Dependable Systems and Networks (DSN 2004), 28 June - 1 July 2004, Florence, Italy, Proceedings*, pages 93–101. IEEE Computer Society, 2004. 127

[KNP13] Sebastian Kutzner, Phuong Ha Nguyen, and Axel Poschmann. Enabling 3-Share Threshold Implementations for all 4-Bit S-Boxes. In Hyang-Sook Lee and Dong-Guk Han, editors, *Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers*, volume 8565 of *Lecture Notes in Computer Science*, pages 91–108. Springer, 2013. 112

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996. 4, 11

[KRJ14] Mohamed Karroumi, Benjamin Richard, and Marc Joye. Addition with Blinded Operands. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2014. 7, 92, 93, 96, 97, 99, 156

[KS73] Peter M. Kogge and Harold S. Stone. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. *IEEE Trans. Comput.*, 22(8):786–793, 1973. 94

[KW13] Ilya Kizhvatov and Marc Witteman. Academic vs. industrial perspective on SCA, and an industrial innovation. *Short talk at COSADE 2013*, 2013. 38

[LB10] Thanh-Ha Le and Maël Berthier. Mutual Information Analysis under the View of Higher-Order Statistics. In Isao Echizen, Noboru Kunihiro, and Ryôichi Sasaki, editors, *Advances in Information and Computer Security - 5th International Workshop on Security, IWSEC 2010, Kobe, Japan, November 22-24, 2010. Proceedings*, volume 6434 of *Lecture Notes in Computer Science*, pages 285–300. Springer, 2010. 66

[Lim98] Chae Hoon Lim. CRYPTON: A New 128-bit Block Cipher - Specification and Analysis. *NIST AES Proposal*, 1998. 106, 117

[Lim99] Chae Hoon Lim. A Revised Version of Crypton - Crypton V1.0. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 1999. 106, 117

[LMW14] Andrew J. Leiserson, Mark E. Marson, and Megan A. Wachs. Gate-Level Masking under a Path-Based Leakage Metric. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 580–597. Springer, 2014. 6, 12, 26, 27

[Low13] Y.M. Low. A new distribution for fitting four moments and its applications to reliability analysis. *Structural Safety*, 42(0):12 – 25, 2013. 70, 71

[LP07] Kerstin Lemke-Rust and Christof Paar. Gaussian Mixture Models for Higher-Order Side Channel Analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 14–27. Springer, 2007. 68

[Mat93] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993. 107

[Mat97] Mitsuru Matsui. New Block Encryption Algorithm MISTY. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 1997. 109

[MGH14] Amir Moradi, Sylvain Guilley, and Annelie Heuser. Detecting Hidden Leakages. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, volume 8479 of *Lecture Notes in Computer Science*, pages 324–342. Springer, 2014. 40

[MH15] Amir Moradi and Gesine Hinterwälder. Side-Channel Security Analysis of Ultra-Low-Power FRAM-Based MCUs. In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 239–254. Springer, 2015. 6, 26

[MI14] Amir Moradi and Vincent Immler. Early Propagation and Imbalanced Routing, How to Diminish in FPGAs. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 598–615. Springer, 2014. 54

[Miy90] Shoji Miyaguchi. The FEAL Cipher Family. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 627–638. Springer, 1990. 7, 92

[MKEP12] Amir Moradi, Mario Kirschbaum, Thomas Eisenbarth, and Christof Paar. Masked dual-rail precharge logic encounters state-of-the-art power analysis methods. *IEEE Trans. VLSI Syst.*, 20(9):1578–1589, 2012. 68

[MM12] Amir Moradi and Oliver Mischke. How Far Should Theory Be from Practice? - Evaluation of a Countermeasure. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2012. 31

[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, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2013. 103

[MME10] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2010. 6, 13, 54, 87, 92

[MOBW13] Luke Mather, Elisabeth Oswald, Joe Bandenburg, and Marcin Wójcik. Does My Device Leak Information? An a priori Statistical Power Analysis of Leakage Detection Tests. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 486–505. Springer, 2013. 26

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards.* Springer, 2007. 13, 91

[Mor12] Amir Moradi. Statistical Tools Flavor Side-Channel Collision Attacks. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 428–445. Springer, 2012. 65

[Mor14a] Amir Moradi. Side-Channel Leakage through Static Power - Should We Care about in Practice? In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 562–579. Springer, 2014. 39

[Mor14b] Amir Moradi. Wire-Tap Codes as Side-Channel Countermeasure - - An FPGA-Based Experiment -. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 341–359. Springer, 2014. 128

[MOS11] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Information Security*, 5(2):100–110, 2011. 65

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005. 13, 64

[MPL$^+$11] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011. 64, 87, 88, 97, 106, 110, 111, 117, 145

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully Attacking Masked AES Hardware Implementations. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005. 5, 8, 13, 87, 92, 106

[MS77] Florence Jessie MacWilliams and N. J. A. Neil James Alexander Sloane. *The Theory of Error Correcting Codes*. North-Holland mathematical library. North-Holland Pub. Co. New York, Amsterdam, New York, 1977. Includes index. 129

[MS14] Amir Moradi and François-Xavier Standaert. Moments-correlating dpa. Cryptology ePrint Archive, Report 2014/409, 2014. `https://eprint.iacr.org/2014/409`. 6, 13, 46, 54, 64, 65, 74, 75, 207

[MS16a] Amir Moradi and Tobias Schneider. Improved Side-Channel Analysis Attacks on Xilinx Bitstream Encryption of 5, 6, and 7 Series. In François-Xavier Standaert and Elisabeth Oswald, editors, *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, volume 9689 of *Lecture Notes in Computer Science*, pages 71–87. Springer, 2016.

[MS16b] 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`.

[MS16c] Amir Moradi and Tobias Schneider. Side-channel analysis protection and low-latency in action - - case study of PRINCE and midori -. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 517–547, 2016.

[MS16d] Amir Moradi and Tobias Schneider. Side-channel analysis protection and low-latency in action - case study of prince and midori. Cryptology ePrint Archive, Report 2016/481, 2016. `https://eprint.iacr.org/2016/481`. 87

[MW15] Amir Moradi and Alexander Wild. Assessment of Hiding the Higher-Order Leakages in Hardware - What Are the Achievements Versus Overheads? In Tim

Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2015. 11, 13, 87, 149

[NIS12] NIST. Secure Hash Standard (SHS) (FIPS PUB 180-4), 2012. 92

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006. 11, 12, 87, 138

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011. 64, 74, 93, 97

[NSGD12] Maxime Nassar, Youssef Souissi, Sylvain Guilley, and Jean-Luc Danger. RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In Wolfgang Rosenstiel and Lothar Thiele, editors, *2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012, Dresden, Germany, March 12-16, 2012*, pages 1173–1178. IEEE, 2012. 40

[Pap16] Panos Papadimitratos. Security on Wheels: Security and Privacy for Vehicular Communication Systems. 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 1855–1856. ACM, 2016. 3

[PCNM15] Sikhar Patranabis, Abhishek Chakraborty, Phuong Ha Nguyen, and Debdeep Mukhopadhyay. A Biased Fault Attack on the Time Redundancy Countermeasure for AES. In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 189–203. Springer, 2015. 130

[Pea95] Karl Pearson. Contributions to the Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material. *Royal Society of London Philosophical Transactions Series A*, 186:343–414, 1895. 69

[Pea01] Karl Pearson. Mathematical Contributions to the Theory of Evolution. X. Supplement to a Memoir on Skew Variation. *Royal Society of London Philosophical Transactions Series A*, 197:443–459, 1901. 69

[Pea16] Karl Pearson. Mathematical Contributions to the Theory of Evolution. XIX. Second Supplement to a Memoir on Skew Variation. *Royal Society of London Philosophical Transactions Series A*, 216:429–457, 1916. 69

[Péb08] Philippe Pébay. Formulas for Robust, One-Pass Parallel Computation of Covariances and Arbitrary-Order Statistical Moments. *Sandia Report SAND2008-6212, Sandia National Laboratories*, 2008. 22, 32, 35, 50

[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. Cryptology*, 24(2):322–345, 2011. 74, 87, 110, 145, 147, 148

[Pos09] Axel York Poschmann. *Lightweight cryptography: cryptographic engineering for a pervasive world.* PhD thesis, Ruhr University Bochum, 2009. 122

[PR11] Emmanuel Prouff and Thomas Roche. Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2011. 131

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009. 14, 35, 51, 65

[PRC12] Gilles Piret, Thomas Roche, and Claude Carlet. PICARO - A Block Cipher Allowing Efficient Higher-Order Side-Channel Resistance. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *Applied Cryptography and Network Security - 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings*, volume 7341 of *Lecture Notes in Computer Science*, pages 311–328. Springer, 2012. 106, 109

[Pub01] NIST FIPS Pub. 197: Advanced encryption standard (aes). *Federal Information Processing Standards Publication*, 197:441–0311, 2001. 8, 143

[Rad04] Håvard Raddum. More Dual Rijndaels. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard - AES, 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers*, volume 3373 of *Lecture Notes in Computer Science*, pages 142–147. Springer, 2004. 112

[RB01] Vincent Rijmen and Paulo S. L. M. Barreto. The WHIRLPOOL hash function. *World-Wide Web document*, page 72, 2001. 117

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating Masking Schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015. 84, 87, 88, 89, 102, 104, 149

[Rep15] Oscar Reparaz. A note on the security of higher-order threshold implementations. Cryptology ePrint Archive, Report 2015/001, 2015. `https://eprint.iacr.org/2015/001`. 41, 43, 102, 103

[RGV12] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting Time Samples for Multivariate DPA Attacks. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 155–174. Springer, 2012. 15, 43

[RGV14] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Generic DPA Attacks: Curse or Blessing? In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*, pages 98–111. Springer, 2014. 81

[RLK11] Thomas Roche, Victor Lomné, and Karim Khalfallah. Combined Fault and Side-Channel Attack on Protected Implementations of AES. In Emmanuel Prouff, editor, *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, volume 7079 of *Lecture Notes in Computer Science*, pages 65–83. Springer, 2011. 128, 135

[RMB$^+$17] Oscar Reparaz, Lauren De Meyer, Begül Bilgin, Victor Arribas, Svetla Nikova, Ventzislav Nikov, and Nigel Smart. Capa: The spirit of beaver against physical attacks. Cryptology ePrint Archive, Report 2017/1195, 2017. `https://eprint.iacr.org/2017/1195`. 157

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010. 123

[RP12] Thomas Roche and Emmanuel Prouff. Higher-order glitch free implementation of the AES using secure multi-party computation protocols - extended version. *J. Cryptographic Engineering*, 2(2):111–127, 2012. 64

[RSDT13] Cyril Roscian, Alexandre Sarafianos, Jean-Max Dutertre, and Assia Tria. Fault Model Analysis of Laser-Induced Faults in SRAM Memory Cells. In Wieland Fischer and Jörn-Marc Schmidt, editors, *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013*, pages 89–98. IEEE Computer Society, 2013. 140

[RSV$^+$11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the*

Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2011. 6, 7, 76

[SAD$^+$16] Meryem Simsek, Adnan Aijaz, Mischa Dohler, Joachim Sachs, and Gerhard Fettweis. 5G-Enabled Tactile Internet. *IEEE Journal on Selected Areas in Communications*, 34(3):460–473, 2016. 3

[Sak] Sakura. Side-channel AttacK User Reference Architecture. `http://satoh.cs.uec.ac.jp/SAKURA/index.html`. 38, 41, 74, 99, 150

[Sco79] David W. Scott. On Optimal and Data-Based Histograms. *Biometrika*, 66(3):605–610, 1979. 67

[SDK$^+$13] Daehyun Strobel, Benedikt Driessen, Timo Kasper, Gregor Leander, David Oswald, Falk Schellenberg, and Christof Paar. Fuming Acid and Cryptanalysis: Handy Tools for Overcoming a Digital Locking and Access Control System. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2013. 4

[SES17] Okan Seker, Thomas Eisenbarth, and Rainer Steinwandt. Extending glitch-free multiparty protocols to resist fault injection attacks. Cryptology ePrint Archive, Report 2017/269, 2017. `https://eprint.iacr.org/2017/269`. 157

[SGS08] Berk Sunar, Gunnar Gaubatz, and Erkay Savas. Sequential Circuit Design for Embedded Cryptographic Applications Resilient to Adversarial Faults. *IEEE Trans. Computers*, 57(1):126–138, 2008. 132

[She04] Simon J. Sheather. Density Estimation. *Statistical Science*, 19(4):588–597, 2004. 67

[Sil86] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986. 67

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005. 66

[SM15a] Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015. 5, 6, 25, 64, 155

[SM15b] Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - a clear roadmap for side-channel evaluations. Cryptology ePrint Archive, Report 2015/207, 2015. `http://eprint.iacr.org/2015/207`.

[SM16] Tobias Schneider and Amir Moradi. Leakage assessment methodology - Extended version. *J. Cryptographic Engineering*, 6(2):85–99, 2016. 5, 6, 25, 49, 155

[SMG15a] Pascal Sasdrich, Amir Moradi, and Tim Güneysu. Affine Equivalence and Its Application to Tightening Threshold Implementations. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566 of *Lecture Notes in Computer Science*, pages 263–276. Springer, 2015. 136, 142, 145, 148

[SMG15b] Tobias Schneider, Amir Moradi, and Tim Güneysu. Arithmetic Addition over Boolean Masking - Towards First- and Second-Order Resistance in Hardware. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, volume 9092 of *Lecture Notes in Computer Science*, pages 559–578. Springer, 2015. 5, 6, 7, 26, 87, 91, 156

[SMG15c] Tobias Schneider, Amir Moradi, and Tim Güneysu. Arithmetic Addition over Boolean Masking - Towards First- and Second-Order Resistance in Hardware. Cryptology ePrint Archive, Report 2015/066, 2015. `http://eprint.iacr.org/2015/066`.

[SMG15d] Tobias Schneider, Amir Moradi, and Tim Güneysu. Robust and One-Pass Parallel Computation of Correlation-Based Attacks at Arbitrary Order. Cryptology ePrint Archive, Report 2015/571, 2015. `http://eprint.iacr.org/2015/571`.

[SMG16a] Tobias Schneider, Amir Moradi, and Tim Güneysu. ParTI – Towards Combined Hardware Countermeasures against Side-Channel and Fault-Injection Attacks. Cryptology ePrint Archive, Report 2016/648, 2016. `http://eprint.iacr.org/2016/648`.

[SMG16b] Tobias Schneider, Amir Moradi, and Tim Güneysu. ParTI - Towards Combined Hardware Countermeasures Against Side-Channel and Fault-Injection Attacks. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 302–332. Springer, 2016. 5, 8, 87, 127, 156

[SMG16c] Tobias Schneider, Amir Moradi, and Tim Güneysu. Robust and One-Pass Parallel Computation of Correlation-Based Attacks at Arbitrary Order. In François-Xavier Standaert and Elisabeth Oswald, editors, *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, volume 9689 of *Lecture Notes in Computer Science*, pages 199–217. Springer, 2016. 5, 6, 19, 45, 155

[SMMG15a] Pascal Sasdrich, Oliver Mischke, Amir Moradi, and Tim Güneysu. Side-Channel Protection by Randomizing Look-Up Tables on Reconfigurable Hardware - Pitfalls of Memory Primitives. In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 95–107. Springer, 2015. 6, 26

[SMMG15b] Pascal Sasdrich, Amir Moradi, Oliver Mischke, and Tim Güneysu. Achieving side-channel protection with dynamic logic reconfiguration on modern FPGAs. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015*, pages 130–136, 2015. 6, 26

[SMSG16a] Tobias Schneider, Amir Moradi, François-Xavier Standaert, and Tim Güneysu. Bridging the gap: Advanced tools for side-channel leakage estimation beyond gaussian templates and histograms. In Roberto Avanzi and Howard M. Heys, editors, *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, volume 10532 of *Lecture Notes in Computer Science*, pages 58–78. Springer, 2016. 63, 155

[SMSG16b] Tobias Schneider, Amir Moradi, François-Xavier Standaert, and Tim Güneysu. Bridging the gap: Advanced tools for side-channel leakage estimation beyond gaussian templates and histograms. Cryptology ePrint Archive, Report 2016/719, 2016. https://eprint.iacr.org/2016/719. 5, 7

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009. 6, 7, 76, 81, 155

[SPR$^+$04] François-Xavier Standaert, Gilles Piret, Gaël Rouvroy, Jean-Jacques Quisquater, and Jean-Didier Legat. ICEBERG : An Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 279–299. Springer, 2004. 106, 117

[SSA$^+$07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In Alex Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2007. 117

[STE15] Aria Shahverdi, Mostafa Taha, and Thomas Eisenbarth. Silent Simon: A threshold implementation under 100 slices. In *IEEE International Symposium on Hardware*

Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015, pages 1–6. IEEE Computer Society, 2015. 87, 110

[SvMG13] Tobias Schneider, Ingo von Maurich, and Tim Güneysu. Efficient implementation of cryptographic primitives on the GA144 multi-core architecture. In *24th International Conference on Application-Specific Systems, Architectures and Processors, ASAP 2013, Washington, DC, USA, June 5-7, 2013*, pages 67–74. IEEE Computer Society, 2013.

[SvMGO14] Tobias Schneider, Ingo von Maurich, Tim Güneysu, and David Oswald. Cryptographic Algorithms on the GA144 Asynchronous Multi-Core Processor - Implementation and Side-Channel Analysis. *Signal Processing Systems*, 77(1-2):151–167, 2014.

[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 Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2010. 14, 35, 51, 68, 82

[TEL15] TELECOM ParisTech. DPA Contest (4th edition), 2013-2015. http://www.DPAcontest.org/v4/. 40

[TV04] Kris Tiri and Ingrid Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*, pages 246–251. IEEE Computer Society, 2004. 128

[UCI+11] Markus Ullrich, Christophe De Cannière, Sebastiaan Indesteege, Özgül Küçük, Nicky Mouha, and Bart Preneel. Finding Optimal Bitsliced Implementations of 4×4-bit S-boxes. In *Symmetric Key Encryption Workshop*, page 20, 2011. 114

[Vir04] Virtual Silicon Inc. 0.18 μm VIP Standard Cell Library Tape Out Ready, Part Number: UMCL18G212T3, Process: UMC Logic 0.18 μm Generic II Technology: 0.18μm, July 2004. 116, 149

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012. 4, 11, 13

[WMG15] Alexander Wild, Amir Moradi, and Tim Güneysu. Evaluating the Duplication of Dual-Rail Precharge Logics on FPGAs. In Stefan Mangard and Axel Y.

Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2015. 6, 11, 13, 26

[WN94] David J. Wheeler and Roger M. Needham. TEA, a Tiny Encryption Algorithm. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 363–366. Springer, 1994. 92

[WOS14] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. The Myth of Generic DPA...and the Magic of Learning. In Josh Benaloh, editor, *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, volume 8366 of *Lecture Notes in Computer Science*, pages 183–205. Springer, 2014. 64, 66, 81

[Wu08] Hongjun Wu. The Stream Cipher HC-128. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 39–47. Springer, 2008. 92

[XGK12] Debdeep Mukhopadhyay Xiaofei Guo and Ramesh Karri. Provably secure concurrent error detection against differential fault analysis. Cryptology ePrint Archive, Report 2012/552, 2012. `https://eprint.iacr.org/2012/552`. 128

List of Abbreviations

AB Almost Bent

AES Advanced Encryption Standard

ANF Algebraic Normal Form

ANSSI Agence nationale de la sécurité des systèmes d'information

APN Almost Perfect Nonlinear

ASIC Application Specific Integrated Circuit

BSI Bundesamt für Sicherheit in der Informationstechnik

CED Concurrent Error Detection

CEPACA Correlation-Enhanced Power Analysis Collision Attack

CPA Correlation Power Analysis

CPU Central Processing Unit

DC Direct Current

DPA Differential Power Analysis

DSO Digital Storage Oscilloscope

DUT Device Under Test

EDC Error Detecting Codes

EMG Exponentially Modified Gaussian

FI Fault Injection

FPGA Field Programmable Gate Array

HD Hamming Distance

HW Hamming Weights

IoT Internet of Things

KSA Kogge-Stone Adder

LFSR Linear Feedback Shift Register

MC-DPA Moments-Correlating DPA

MCC-DPA Moments Correlating Collision-DPA

MCP-DPA Moments-Correlating Profiled DPA

MI Mutual Information

MIA Mutual Information Analysis

μC Microcontroller

NLFSR Non-Linear Feedback Shift Register

PDF Probability Density Function

PI Perceived Information

PRNG Pseudo-Random Number Generator

RCA Ripple Carry Adder

SCA Side-Channel Analysis

SGL Shifted Generalized Lognormal

SNR Signal-to-Noise Ratio

SPN Substitution-Permutation Network

TA Template Attack

TI Threshold Implementation

XOR Exclusive OR

List of Figures

3.1 The influence of the first four statistical moments on the location, spread, and shape of a distribution. 20

4.1 Student's t distribution functions and two-tailed Welch's t-test (examples for $v = 10,000$). 28
4.2 An optimized measurement process. 39
4.3 DPA contest v4.2, non-specific t-test results (top) first-order, (bottom) second-order univariate using $100,000$ traces. 42
4.4 Architecture of the second-order TI of the NLFSR. 42
4.5 NLFSR 2nd-order TI, sample trace and univariate and bivariate non-specific t-test results using $2,000,000$ traces. 44

5.1 Difference between the result of correlation estimations (raw-moment versus three-pass). 60

6.1 Summary of side-channel evaluation tools and attacks. 65
6.2 Plane of existence of the different distributions. 72
6.3 The estimated moments for each possible $s \in \{0,1\}^4$ (a) and kernel-estimated PDFs (b) for mathematically-generated leakages corresponding to a 2nd-order masking. 73
6.4 The estimated PDFs for mathematically-generated leakages corresponding to a 2nd-order masking, obtained with various parametric tools from Sections 6.2 and 6.3. 73
6.5 (a) sample trace. (b) first-order, (c) second-order, and (d) third-order MCP-DPA results for different time samples in the leakage traces (taken from [MS14]). . . . 75
6.6 Kernel-, Gaussian-, EMG-, Pearson- and SGL-based PI estimation with all covered moments (a) using 100,000,000 meas., (b) over the number of meas. 77
6.7 The PDFs of the six distributions from Table 6.1 and 6.2. 78
6.8 The estimated PDFs of the four distributions from Table 6.1 with partly fixed moments according to the four evaluations cases. 79
6.9 PI estimates for the separate moments. 81
6.10 Success rate of several univariate template attacks exploiting separate and combined moments, for the most informative sample point 719 in our traces. 82
6.11 Guessing entropy for MIA based on different estimation tools (at sample 719). . 83
6.12 Additional (profiled and non-profiled) attacks with kernel density estimation and comparison with other attacks exploiting all the moments at time sample 719. . . 84

8.1 The structure of the carry generation for 4-bit operands using the KSA 94
8.2 Structure of the first-order secure adder based on RCA. 96

8.3 Block diagram of the first-order secure adder based on KSA. 98
8.4 RCA 1st order, t-test results using 100 000 000 traces. 100
8.5 RCA 2nd order, t-test results using 300 000 000 traces. 101
8.6 KSA 1st order, t-test results using 100 000 000 traces. 101
8.7 KSA 2nd order, t-test results using 300 000 000 traces. 102
8.8 (modified measurement setup) KSA 2nd order, t-test results using 300 000 000 traces. 103

9.1 (a): Feistel (b) MISTY (c) SPN 109
9.2 Different serialized design architectures. 111
9.3 Structure of the Sboxes of CLEFIA, Crypton V0.5, Crypton V1, and ICEBERG. 112
9.4 Structure of the Sboxes of Fantomas, Khazad, and Whirlpool. 114
9.5 Threshold Implementation of Robin and Fantomas Sboxes, each signal represents 3 shares, the gray registers for pipeline variants. 116
9.6 Illustration of the two equivalence representation of Feistel^1_G. 120
9.7 The smallest achievable differential uniformity and linearity for each number of iterations for round functions with $\mathbb{F}_{16}$-linear layers and $F_1 = \mathcal{A}^4_0$ and $(\diamond)F_2 = \mathcal{Q}^4_4$, $(*)F_2 = \mathcal{Q}^4_{12}$, $(\triangle)F_2 = \mathcal{Q}^4_{293}$, $(\circ)F_2 = \mathcal{Q}^4_{294}$, $(\Box)F_2 = \mathcal{Q}^4_{299}$. 121
9.8 Bitslice masked implementation for the ATMEGA644p. 124

10.1 A common structure of CED schemes using EDC. 131
10.2 The basic structure of our combined scheme. 134
10.3 Derivation of shared predictors for three shares. 136
10.4 Computation and unmasking of the error check vector for three shares in a first-order secure design. 137
10.5 The basic structure of LED-128. 144
10.6 The basic structure of our proposed LED design. Multiplexers for the plaintext and AddRoundKey are omitted. 149
10.7 A sample trace and the result of non-specific t-tests at orders one to three. 151

List of Tables

4.1 Comparison of the accuracy of different methods to compute the second parameter of the t-tests (100 million simulated traces $\sim \mathcal{N}(100, 25)$). 34

6.1 The first four statistical moments of four distributions at sample point 719. . . . 78
6.2 The first four statistical moments of two simulated distributions. 80

8.1 Results and comparison of our hardware architectures. 99

9.1 Performance figures of 4×4 quadratic bijections with respect to their TI cost. . 110
9.2 Criteria for the 8-bit Sboxes. 117
9.3 Specifics of the selected Sboxes. 123

10.1 Size of our design for an ASIC platform. 150
10.2 Fault coverage for different distributions and codes. 152

14.1 Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_1$. 173
14.2 Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_2$. 173
14.3 Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_3$. 173
14.4 Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_4$. 173
14.5 Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_5$. 173
14.6 Distribution of the algebraic degrees of the component functions of $\mathbf{SB}_6$. 173
14.7 The round function of $\mathbf{SB}_1(xy)$. 8 iterations of it result in the final Sbox. 174
14.8 The round function of $\mathbf{SB}_2(xy)$. 2 iterations of it result in the final Sbox. 174
14.9 The round function of $\mathbf{SB}_3(xy)$. 4 iterations of it result in the final Sbox. 175
14.10 The round function of $\mathbf{SB}_4(xy)$. 5 iterations of it result in the final Sbox. 175
14.11 The round function of $\mathbf{SB}_5(xy)$. 9 iterations of it result in the final Sbox. 176
14.12 The round function of $\mathbf{SB}_6(xy)$. 4 iterations of it result in the final Sbox. 176

About the Author

Author information as of October 2016.

Personal Data

Name Tobias Schneider

Address
Chair for Embedded Security, ID 2/637,
Universitätsstr. 150,
44801 Bochum, Germany

E-Mail tobias.schneider-a7a@rub.de

Date of birth January 27, 1990

Place of birth Oberhausen, Germany

Education

10/2013 - 02/2017	**Dr.-Ing.**, *Ruhr-Universität Bochum*, Electrical and Information Engineering. Fast-Track PhD (TopING)
10/2012 - 03/2015	**M.Sc.**, *Ruhr-Universität Bochum*, IT Security/Information Engineering. Average Score: Excellent (100%)
10/2009 - 09/2012	**B.Sc.**, *Ruhr-Universität Bochum*, IT Security/Information Engineering. Average Score: Excellent (95%)

Professional Experience

10/2013 - 08/2017	**Research Assistant**, *Ruhr-Universität Bochum.* Chair for Embedded Security (EmSec)
07/2012 - 09/2012	**Intern**, *ESCRYPT*, Bochum. Development of Side-Channel Analysis Framework

Publications and Academic Activities

Peer-Reviewed Journal Papers

- Tobias Schneider, Ingo von Maurich, Tim Güneysu, and David Oswald. Cryptographic Algorithms on the GA144 Asynchronous Multi-Core Processor - Implementation and Side-Channel Analysis. *Signal Processing Systems*, 77(1-2):151–167, 2014

- Tobias Schneider and Amir Moradi. Leakage assessment methodology - Extended version. *J. Cryptographic Engineering*, 6(2):85–99, 2016

Peer-Reviewed Conference Proceeding

- Tobias Schneider, Ingo von Maurich, and Tim Güneysu. Efficient implementation of cryptographic primitives on the GA144 multi-core architecture. In *24th International Conference on Application-Specific Systems, Architectures and Processors, ASAP 2013, Washington, DC, USA, June 5-7, 2013*, pages 67–74. IEEE Computer Society, 2013

- Tobias Schneider, Amir Moradi, and Tim Güneysu. Arithmetic Addition over Boolean Masking - Towards First- and Second-Order Resistance in Hardware. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, volume 9092 of *Lecture Notes in Computer Science*, pages 559–578. Springer, 2015

- Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015

- Tobias Schneider, Amir Moradi, and Tim Güneysu. Robust and One-Pass Parallel Computation of Correlation-Based Attacks at Arbitrary Order. In François-Xavier Standaert and Elisabeth Oswald, editors, *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, volume 9689 of *Lecture Notes in Computer Science*, pages 199–217. Springer, 2016

- Amir Moradi and Tobias Schneider. Improved Side-Channel Analysis Attacks on Xilinx Bitstream Encryption of 5, 6, and 7 Series. In François-Xavier Standaert and Elisabeth Oswald, editors, *Constructive Side-Channel Analysis and Secure Design - 7th International*

Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers, volume 9689 of *Lecture Notes in Computer Science*, pages 71–87. Springer, 2016

- Erik Boss, Vincent Grosso, Tim Güneysu, Gregor Leander, Amir Moradi, and Tobias Schneider. Strong 8-bit Sboxes with Efficient Masking in Hardware. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 171–193. Springer, 2016
- Tobias Schneider, Amir Moradi, and Tim Güneysu. ParTI - Towards Combined Hardware Countermeasures Against Side-Channel and Fault-Injection Attacks. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 302–332. Springer, 2016
- Tobias Schneider, Amir Moradi, François-Xavier Standaert, and Tim Güneysu. Bridging the gap: Advanced tools for side-channel leakage estimation beyond gaussian templates and histograms. In Roberto Avanzi and Howard M. Heys, editors, *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, volume 10532 of *Lecture Notes in Computer Science*, pages 58–78. Springer, 2016
- Amir Moradi and Tobias Schneider. Side-channel analysis protection and low-latency in action - - case study of PRINCE and midori -. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 517–547, 2016

Technical Reports

- Tobias Schneider, Amir Moradi, and Tim Güneysu. Arithmetic Addition over Boolean Masking - Towards First- and Second-Order Resistance in Hardware. Cryptology ePrint Archive, Report 2015/066, 2015. `http://eprint.iacr.org/2015/066`
- Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - a clear roadmap for side-channel evaluations. Cryptology ePrint Archive, Report 2015/207, 2015. `http://eprint.iacr.org/2015/207`
- Tobias Schneider, Amir Moradi, and Tim Güneysu. Robust and One-Pass Parallel Computation of Correlation-Based Attacks at Arbitrary Order. Cryptology ePrint Archive, Report 2015/571, 2015. `http://eprint.iacr.org/2015/571`
- 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`

- Erik Boss, Vincent Grosso, Tim Güneysu, Gregor Leander, Amir Moradi, and Tobias Schneider. Strong 8-bit Sboxes with Efficient Masking in Hardware. Cryptology ePrint Archive, Report 2016/647, 2016. `http://eprint.iacr.org/2016/647`
- Tobias Schneider, Amir Moradi, and Tim Güneysu. ParTI – Towards Combined Hardware Countermeasures against Side-Channel and Fault-Injection Attacks. Cryptology ePrint Archive, Report 2016/648, 2016. `http://eprint.iacr.org/2016/648`
- Tobias Schneider, Amir Moradi, FranÃğois-Xavier Standaert, and Tim GÃijneysu. Bridging the gap: Advanced tools for side-channel leakage estimation beyond gaussian templates and histograms. Cryptology ePrint Archive, Report 2016/719, 2016. `https://eprint.iacr.org/2016/719`
- Amir Moradi and Tobias Schneider. Side-channel analysis protection and low-latency in action - case study of prince and midori. Cryptology ePrint Archive, Report 2016/481, 2016. `https://eprint.iacr.org/2016/481`

Invited Talks

- Leakage assessment methodology - a clear roadmap for side-channel evaluations. *Workshop on Implementation: Security and Evaluation (WISE), 11 September 2015, Paris, France*

Awards and Stipends

01/2016	**Faculty Award** - Best M.Sc. graduate in IT security (RUB).
Since 10/2013	**Member TopING** - Fast-Track Phd (RUB).
01/2013	**G Data Award** - Two best B.Sc. graduates in IT security (RUB).
10/2012 - 09/2013	**Qualifying Fellowship** - Ubicrypt (DFG-GRK 1817).

Participation in Selected Conferences, Workshops and Summer Schools

- CRYPTO 2016, *Santa Barbara, USA*
- CHES 2016, *Santa Barbara, USA*
- WhibOx 2016, *Santa Barbara, USA*
- SAC 2016, *St. John's, Canada*
- DISC 2016, *Bochum, Germany*

- FSE 2016, *Bochum, Germany*
- Spring School on Symmetric Cryptography 2016, *Bochum, Germany*
- CHES 2015, *Saint-Malo, France*
- WISE 2015, *Paris, France*
- TI day 2015, *Leuven, Belgium*
- ACNS 2015, *New York, USA*
- Summer School on Design and security of cryptographic algorithms and devices for real-world applications, *Šibenik, Croatia*
- 4th Bar-Ilan Winter School on Cryptography, *Tel Aviv, Israel*
- ASAP 2013, *Washington D.C., USA*