KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

New Security Models for Passive and Active Physical Attacks

Siemen Dhooghe

Supervisors:
Prof. dr. ir. Vincent Rijmen
Dr. Svetla Nikova

Dissertation presented in partial fulfilment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

December 2021

New Security Models for Passive and Active Physical Attacks

Siemen DHOOGHE

Examination committee:
Em. prof. dr. ir. Paul Sas, chair
Prof. dr. ir. Vincent Rijmen, supervisor
Dr. Svetla Nikova, supervisor
Prof. dr. Nigel Smart
Prof. dr. ir. Luc Van Eycken
Dr. Ventzislav Nikov
(NXP Semiconductors)
Dr. Emmanuel Prouff
(ANSSI, France)
Prof. dr. François-Xavier Standaert
(Université Catholique de Louvain)

Dissertation presented in partial fulfilment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

December 2021

© 2021 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Siemen Dhooghe, Kasteelpark Arenberg 10, bus 2452, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Acknowledgements

> To make a prairie it takes a clover
> and one bee,
> One clover, and a bee.
> And revery.
> The revery alone will do,
> If bees are few.
>
> Emily Dickinson

I would not be here without the help of so many teachers, colleagues, friends, and family. For that reason I want to dedicate this work to them. I will not be able to list everyone, but a small subset is given special acknowledgement below.

First and foremost, I want to thank my supervisors Svetla and Vincent. In all my time as a PhD student, you have always trusted and supported me. No matter my achievements, you stood behind me. This support has been nothing short of inspirational.

Next, I want to thank Ventzi for all the interesting discussions and for keeping my feet on the ground. Because of you I understand the practice of the field.

I want to thank some of my teachers without whom I would not be here. Natasha Raymaekers for being one of the most amazing maths teachers a high school could hope for, without you we would not even have been able to enter university. Wim Veys for showing me the beauty of linear algebra. The physics lessons paled compared to yours. Gilles Zémor and Karim Belabas for introducing me to the interesting field of cryptography and supporting me to enter engineering.

I want to thank Tomer and Alan. You were the first people I worked with. I could not wish for better role models. You shaped me into the researcher I am

today.

I want to thank Tim. Working together with you gave me such an amazing insight in the field but more than that, it has been a lot of fun. To find someone who has the same idea about cryptography in a field like side-channel analysis is rare indeed.

I want to thank Adrián and Liliya who have been my colleagues, office mates, and friends. I will never forget all the hours we shared at a white board or all the hours we spend laughing next to it.

I want to thank Gabriele who has helped me through my first year and taught me, a lost mathematician, all the basics of hardware.

A lot of thanks to my housemates Ankatrien, Dana, Dennis, Fleury, Foster, Jona, Jonas, and Rodrigo for providing such a fun place to live in these last four years. No matter the stress at work, I knew there were people waiting for me at home.

My thanks to Wendy. People sometimes get stuck in their own mind. Talking to you was like having a map to navigate through it.

I want to thank all my amazing friends. Lien for all the walks and the sharing of PhD experiences. Carole for all the rants and laughs. Katrijne for always keeping us together. Matthias for being my sensei in PhD's and cycling.

Then there is Milan. I was born an only child but was so lucky to find a brother. Words cannot describe, but with you they are also not needed.

I want to thank my parents. You have provided me with everything but most of all you gave me my ambition for mathematics. You have always given me freedom to pursue my own life and even when things were tough you kept supporting me no matter what. I am so lucky to have you.

Finally, thanks to the Fonds Wetenschappelijk Onderzoek (FWO) and the National Institute of Standards and Technology (NIST) for their generous support providing me the means to live and research.

Abstract

Symmetric key cryptography has been the corner stone in defending applications against cyber attacks. Ever since the proposal of the Data Encryption Standard (DES), people were able to protect their applications using top of the line cryptography. Nevertheless, even the top of the line symmetric cryptography is endangered when placed in an embedded device.

In 1999, an attack called *differential power analysis* was made public to the world. The attack abuses the relation between the power consumption of a device and the data it is processing. Using statistical methods, it was shown possible to break DES using only a thousand calls to the primitive. Since the attack was so potent, it became obvious that immediate countermeasures were needed. The attack started a new field named *side-channel analysis*.

One countermeasure withstood the test of time and became a centre piece in the protection of hardware applications. This was *masking*. Masking is essentially an algorithmic countermeasure splitting a cryptographic primitive in multiple parts. Each part seemingly operates on pure random data. However, when the parts are combined, the regular computation of the primitive can be seen again. The essential idea of masking is based on *noise amplification*. Observing multiple parts of the computation at the same time exponentially increases the noise, which was present due to measuring errors or noise inherent to the device. This allows industry to create countermeasures where attackers would need several millions if not billions of power samples in order to break the device. Numbers which are not possible to obtain in practice.

Nevertheless, while masking provides practical protection, it comes with additional hardware costs such as latency and area which is often inhibiting for industry. Also in the academic world the understanding of masking is limited. While there are some security models able to verify the masking, these security models are still crude. The models are too strong assuming the adversary has unlimited resources or queries to the primitive. At the same time, the models

are also too weak as they don't include all the physical effects which occur in practice. In short, those security models do not properly reflect reality and this comes at the cost of potentially insecure maskings or a significant overhead in cost.

In this thesis, we question the perfect security of maskings and instead propose a model where the adversary only has some limited queries to the primitive. We propose an analysis method which allows us to verify the masking of a primitive in its entirety as opposed to methods which verify the security modularly. The analysis method is interesting on two accounts. The first is that it combines the field of side-channel analysis with symmetric key cryptanalysis. This shows a connection between the masking of a primitive and the primitive proper. The second relates to *threshold implementations* and the possibility to reduce the randomness cost of maskings. This cost has often been under-reported and under-investigated. Instead, with the new security model, we find that we can create higher-order threshold implementations, the security of which has been a long-standing open problem. This results in maskings which do not rely on any fresh randomness to secure its operation. We investigate such masking methods further to remove this randomness cost without shifting the cost somewhere else.

The second research direction of the thesis concerns fault analysis. In 1997 Biham and Shamir showed that using a well-placed fault, it becomes possible to attack symmetric ciphers and retrieve their secret key. The authors showcased their attack on DES using only a few faulty ciphertexts. Fault attacks and their countermeasures have been largely under-studied. In this thesis, we propose a simple active adversary and study security models in order to verify the security of countermeasures. More specifically, we propose composable security models such that it becomes possible to secure any primitive. We also extend the analysis to combined attacks where adversaries can use both side-channel analysis and fault attacks. We then investigate extensions to the adversary in order to better capture realistic fault attacks and observe whether the composable security model still works with the new adversary. The result is a first step towards provable secure countermeasures against active and combined attacks.

Beknopte Samenvatting

Symmetrische cryptografie is de hoeksteen in het verdedigen van technologie tegen cyber aanvallen. Sinds het voorstel van de Data Encryptie Standaard (DES) konden mensen beveiligde software en hardware maken. Desalniettemin, deze hoogtechnologische cryptografie beloopt gevaar wanneer het zich bevindt in een ingebed apparaat.

In 1999 werd een aanval genaamd *differentiële vermogen analyse* gepubliceerd. De aanval misbruikt de relatie tussen het vermogen gebruik van een apparaat en de data die daardoor vloeit. Gebruik makende van statistische methoden konden de auteurs de DES breken met maar enkele duizenden keren het cijfer op te roepen. Omdat de aanval zo sterk was werd het meteen duidelijk dat er onmiddellijk tegenmaatregelen nodig waren. Het begon ook een nieuw veld genaamd *nevenkanaal analyse*.

Één maatregel blonk uit tegenover de anderen en werd al vlug het mantelstuk om hardware applicaties te verdedigen. Deze maatregel noemt men *maskeringsmethoden*. De methode splitst essentieel het symmetrische cijfer op in enkele stukken. Elk stuk appart lijkt te werken op willekeurige data, maar breng de stukken bij elkaar en het originele cijfer duikt terug op. Het idee achter deze maskeringmethoden bevindt zich in *ruis amplificatie*. Het observeren van meerdere stukjes van een berekening tegelijkertijd maakt de ruis, die afkomstig was van de meetapparatuur of van het apparaat zelf, exponentieel groter. Dit maakt het voor de industrie mogelijk om een tegenmaatregel te ontwerpen die enkel breekt als je het gemaskeerde cijfer miljoenen of zelfs meer keer moet oproepen. Deze getallen kan je typisch niet bereiken in de praktijk.

Desalniettemin, ook al bereiken maskeringsmethoden praktische veiligheid, ze komen met extra kosten voor hardware. Typisch hebben maskeringsmethode een verlaagde snelheid en een verhoogde oppervlakte kost. Deze extra kosten zijn vaak limiterend voor de industrie. Ook de academische wereld heeft hun zinnen gezet op de maskeringsmethoden. Al zijn er beveiligingsmodellen om

de veiligheid van de maskeringsmethoden te verifiëren, ze zijn nog steeds ruw. Vaak zijn deze modellen te sterk. Bijvoorbeeld, ze beschouwen tegenstrevers met ongelimiteerde grondstoffen of oproepen aan het symmetrische primitief. De modellen zijn ook vaak te zwak omdat ze niet al de fysieke effecten van hardware beschouwen. In het kort, de modellen staan niet dicht genoeg bij de realiteit. Dit resulteerd dan in het maken van onveilige maskeringen of maskeringen die veel te duur zijn.

We bekijken of de perfecte veiligheid van een maskering wel noodzakelijk is. We stellen een beveiligingsmodel voor waarbij een tegenstrever maar een eindig aantal oproepen heeft voor het symmetrisch primitief. We stellen ook een analyse methode voor in dat model waarmee we een maskering in zijn geheel kunnen verifiëren in de plaats het gebruiken van een modulaire methode. Deze analyse is interessant voor twee redenen. De eerste is dat deze de velden van symmetrische sleutel cryptografie en nevenkanaal analyse combineert. Dat legt een connectie bloot tussen het gemaskeerde primitief en het primitief zelf. De tweede reden staat in verband met zogenaamde *drempel implementaties* en de mogelijkheid om de kost van de willekeurigheid in de maskering te reduceren. Deze laatste kost is het aantal cryptografische willekeurige waarden die er moeten gegenereerd worden. De kost is vaak onder gerapporteerd en niet bestudeerd. Met onze nieuwe analyse methode kunnen we hogere-orde drempel implementaties maken, een academisch vraagstuk die al enkele jaren open is gebleven. Deze nieuwe drempel implementaties laten ons toe om maskeringen te maken die het minimum aan extra willekeurige waarden gebruikt. We bestuderen deze maskeringen zodat we deze willekeurigheidskost kunnen verwijderen zonder ze te verplaatsen naar andere kosten.

De tweede onderzoeksrichting draait rond foutenanalyse. In 1997 hebben Biham en Shamir getoond dat een goed geplaatste fout een cijfer zoals DES kan breken. Erger nog, men kan de geheime sleutel van DES terug vinden met maar enkele zulke fouten. Foutenanalyse en zijn tegenmaatregelen zijn grotendeels nog niet onderzocht geweest. In deze thesis stellen we een tegenstrever voor die fouten kan maken in een cijfer. We bestuderen dan beveiligingsmodellen voor deze tegenstrever. Specifieker nog, we geven een composeerbaar model wat betekend dat de compositie tussen circuits die veilig zijn in het model terug veilig is. Hiermee kunnen we tegenmaatregelen maken die elk primitief kan verdedigen. We bereiden dan onze analyse uit om gecombineerde aanvallen te begrijpen. Dit zijn aanvallen waarbij de tegenstrever gebruikt maakt van nevenkanaal analyse en foutenanalyse. We bereiden ook de tegenstrever uit en beschouwen extensies aan de fouten die deze kan zetten zodat die dichterbij de praktijk komt te staan. Alweer bekijken we of we beveiligingsmodellen kunnen maken voor deze sterkere tegenstrever. Het resultaat is een eerste stap naar het maken van bewijsbaar veilige tegenmaatregelen.

Glossary

Activity Pattern A graphical representation showing which S-boxes can be active when placing input and output masks. In this work, we consider only activity patterns for linear approximations.

AES Advanced Encryption Standard. The standard block cipher.

ASIC Application Specific Integrated Circuit.

Bounded-Query Probing Model See Section 2.6.3. A security model for the probing adversary where the adversary only has a limited number of queries.

Changing of the Guards See Section 4.6.2. A technique to make a sharing uniform using other shares in the state.

CMS Consolidating Masking Schemes. See Section 7.4. A masking technique to guarantee second-order security.

Correctness See Definition 4.2. A masked function is correct if the sum of its output shares is the function applied to the sum of its inputs shares.

FPGA Field Programmable Gate Array. Re-programmable hardware.

GE Gate Equivalent. A unit of measurement for the area of the implementation independent of the technology.

Glitches See Section 2.5.3. Uncertainty in race-conditions makes adversaries view different values from the expected digital values. Glitches are stopped by registers.

LFSR Linear Feedback Shift Register. A linear system sometimes used to generate non-cryptographic randomness.

Masking See Definition 2.1. A technique where a value is split in multiple parts using randomness. This thesis only considers Boolean masking.

(d, k)-**NINA** See Section 9.5. A security model that provides composable security against combined (passive and active) adversaries. The parameter d stands for the passive threshold and k for the active.

Non-Accumulation See Definition 9.4. Composable active security model.

Non-Completeness See Definition 4.3 or Definition 5.5. A masked function is non-complete if no coordinate function works on all input shares.

Non-Interference See Definition 2.5. Composable passive security model.

Probing Model See Section 2.5 and Section 2.6. Here the adversary is allowed to view the computation on a threshold number of hardware resources. Several security models have been built around this adversary.

Resilient Uniform Masking See Definition 7.3. A masking is resilient uniform if some shares are fixed and the others uniform. Which shares can be fixed is determined by an adversary structure.

Resilient Uniformity See Definition 7.4. A masked function is resilient uniform if it maps a resilient input masking to a resilient output masking of some adversary structure.

RNG Random Number Generator.

Simulability See Definition 2.4 or Definition 9.2. The security framework for simulation-based proofs for probing or faulting adversaries.

Statistical Ineffective Fault Analysis An injected fault is used to determine the secret state of a device by observing the state of the output. In case a successful fault is injected but the state of the output does not change, the adversary could potentially discover secret information.

Threshold Implementation See Section 4.2. A masked design which fulfils the non-completeness and uniformity properties.

Trace The power consumption of one execution of a device or design. Typically, the data is noisy requiring multiple traces for a usable analysis.

Transition Leakage See Section 2.5.4. Leakage due to the power consumption being related to both the previous as current digital values flowing through the resource.

Uniform Masking See Definition 4.1. A Boolean masking is uniform if all but one of the shares are of full entropy.

Uniformity See Definition 4.4. A function is uniform if its output is a uniform masking given that the input was uniform.

Contents

Acknowledgements i

Abstract iii

Beknopte Samenvatting v

Glossary vii

Contents xi

List of Figures xv

List of Tables xvii

1 Introduction 1
- 1.1 The Context of the Thesis . . . 6
- 1.2 Research Directions . . . 7
- 1.3 Road Map of the Thesis . . . 9

2 Masking as a Countermeasure 13
- 2.1 Motivation . . . 13
- 2.2 Introduction to Hardware . . . 14
- 2.3 Boolean Masking . . . 15
- 2.4 Intuition on Security . . . 16
- 2.5 The Probing Adversary Model . . . 18
  - 2.5.1 The Circuit Model . . . 18
  - 2.5.2 The Probing Adversary Model . . . 19
  - 2.5.3 Modelling Glitches . . . 19
  - 2.5.4 Transition Leakage . . . 20
- 2.6 Security Models for the Probing Model . . . 21
  - 2.6.1 The Probing Security Model . . . 22

2.6.2 The Non-Interference Framework . . . 22
2.6.3 The Bounded-Query Model . . . 24
2.7 Examples of Masked Operations . . . 25
2.7.1 Masked XOR . . . 25
2.7.2 Masked AND . . . 26
2.8 Conclusion . . . 28

3 Generating Randomness in Maskings **29**
3.1 The Cost of Composable Security . . . 29
3.2 Probing Robust Random Generators . . . 30
3.3 Outline of Non-Deterministic Random Number Generation . . . 32
3.4 Generation Using Cryptographic Primitives . . . 34
3.5 Generation Using LFSRs: Three Attacks . . . 35
3.5.1 Attack 1: Direct Probing . . . 36
3.5.2 Attack 2: Misalignment . . . 37
3.5.3 Attack 3: State Retrieval . . . 38
3.6 Conclusion . . . 39

4 Threshold Implementations **41**
4.1 Motivation . . . 41
4.2 Definitions . . . 42
4.3 Security in the Probing Model . . . 44
4.4 Reducing Randomness . . . 45
4.5 The Necessity of Uniformity . . . 48
4.6 Application to AES . . . 49
4.6.1 AES . . . 50
4.6.2 Changing of the Guards With Randomness . . . 50
4.6.3 S-Box Sharing . . . 52
4.7 Lack of Higher-Order Security . . . 53

5 Higher-Order Threshold Implementations **55**
5.1 The Historical Context . . . 55
5.2 Fourier Analysis . . . 56
5.3 Bound on the Advantage . . . 58
5.4 Fourier Analysis of Shared Functions . . . 59
5.4.1 Restrictions of Shared Functions . . . 59
5.4.2 Correlations between Probed Values . . . 60
5.5 Higher-Order Non-Completeness . . . 62
5.6 Application to LED . . . 63
5.6.1 Description of LED . . . 63
5.6.2 Sharing Second-Order LED . . . 64
5.6.3 Probing Security of One Round . . . 65
5.6.4 Probing Nearby Rounds: Zero Correlation . . . 66

5.6.5 Five Rounds or More: Low Correlation 67
5.6.6 Influence of the Key-Schedule 70
5.7 The Remaining Engineering Challenges 71

6 Efficient Higher-Order Threshold implementations 73
6.1 Context 73
6.2 A Noisy Probing Model 74
6.2.1 Noisy Leakage Functions 75
6.2.2 Bound on the Advantage 76
6.2.3 Amortisation of Randomness 77
6.3 Masking Techniques 79
6.3.1 Technique 1: Non-Completeness over Two Stages 79
6.3.2 Technique 2: Paired Masked S-boxes 80
6.4 Application to LED 82
6.4.1 A Low-Latency Variant 82
6.4.2 A Randomness-Free Variant 85
6.5 Application to AES 87
6.5.1 Masking 87
6.5.2 Optimising the S-Box Sharing 89
6.5.3 Guards in Formation 90
6.5.4 Security Analysis 92
6.5.5 Sharpening the Glitch Model 96
6.5.6 Amortising Randomness 98
6.6 Conclusion 99

7 Resilient Uniformity 101
7.1 Motivation 101
7.2 Adversary Structures and Resiliency 102
7.3 Resilient Uniformity 103
7.3.1 Proving Second-Order Probing Security 104
7.4 Consolidating Masking Schemes Revisited 109
7.4.1 The CMS Approach 109
7.4.2 Applying Resilient Uniformity 109
7.5 Conclusion 112

8 Threshold Implementations Against Advanced Leakage 113
8.1 Analysing Transition Leakage 113
8.1.1 PRESENT 114
8.1.2 AES 117
8.1.3 ASCON 120
8.2 Analysing Coupling Leakage 121
8.3 Analysing Glitches, Transitions, and Couplings 122
8.3.1 PRESENT: Serial Architecture 122

8.3.2 AES: Serial Architecture . . . 123
8.3.3 ASCON: Round-Based Architecture . . . 123
8.4 Conclusion . . . 124

9 Fault Injection and Countermeasures 127
9.1 Motivation . . . 127
9.2 Quick Overview of Fault Attacks . . . 128
9.2.1 Fault Injection Mechanisms . . . 128
9.2.2 Breaking Cryptographic Assumptions . . . 129
9.3 Linear Codes and Redundancy . . . 130
9.4 The Adversary and Security Model . . . 130
9.4.1 The Combined Adversary and Security Models . . . 130
9.5 The NINA Security Model . . . 131
9.5.1 The Simulation Model . . . 131
9.5.2 Non-Accumulation . . . 133
9.5.3 NINA . . . 134
9.5.4 A Combined Secure Multiplier . . . 137
9.5.5 Combined Security of the Multiplication . . . 138
9.6 The Tile Model . . . 144
9.6.1 Definition . . . 144
9.6.2 Area Faults . . . 145
9.6.3 Permanent Faults . . . 146
9.6.4 A Combined Secure Tiled Multiplier . . . 147
9.7 Conclusion . . . 148

10 Conclusions & Open Problems 151
10.1 Summary . . . 151
10.2 Directions for Future Work . . . 153

Bibliography 159

List of Figures

1.1 Alice and Bob communicating via a symmetric-key cipher. 1
1.2 Alice performing one of two operations on a computer. 3

2.1 Abstract depiction of a register. 15
2.2 Depiction of an insecure first-order masked AND. 18
2.3 The bounded-query probing model. 25
2.4 Depiction of a first-order masked XOR. 26
2.5 Depiction of a first-order masked AND. 27

3.1 A second-order secure AND gate. 31
3.2 Counter mode of operation. 34
3.3 Description of an LFSR. 36
3.4 First-order sharing of De Meyer *et al.* [47]. 38
3.5 Second-order sharing of De Meyer *et al.* [47]. 38

4.1 Representation of a threshold circuit. 42
4.2 Feistel structure of $\mathcal{Q}_{294}^4$. 47
4.3 Representation of the AES. 50
4.4 Changing of the guards with randomness. 51
4.5 Representation of the first-order AES S-box. 52

5.1 Two probes giving the observation $\mathbf{z} = (\mathbf{x}_I, \mathbf{y}_J)$. 61
5.2 The step function of LED. 63
5.3 One round of masked LED. 64
5.4 Activity patterns for masked LED. 68

6.1 Illustration of the l-block 2-noisy-threshold probing model. . . 78
6.2 Two masked S-boxes sharing the same randomness. 80
6.3 Two paired masked S-boxes. 81
6.4 Extraction of randomness from a masked LED. 86
6.5 One round of the second-order masked AES. 88

6.6 Sharing of the cubic function F (or G). 89
6.7 Diffusion patterns in the second-order AES. 91
6.8 Depiction of the consecutive $\bar{F}$ and $\bar{G}$ functions. 93
6.9 An optimal trail through the second-order masked AES. 95
6.10 Sorted absolute Walsh-spectrum and 1-norm of glitch functions. 97
6.11 Extraction of randomness from a masked AES. 98

7.1 The CMS secure AND gate and refreshing method. 110
7.2 A depiction of the CMS approach. 110

8.1 An activity pattern caused by transition leakage in PRESENT. 116
8.2 Transition leakage in a divided S-box. 116
8.3 Activity patterns between the top left S-boxes of a masked AES. 118
8.4 Transition leakage following the AES diffusion layers. 119
8.5 Diffusion patterns using the changing of the guards technique. . 119
8.6 The round function of ASCON. 120
8.7 Transition leakage in ASCON. 121
8.8 Separation into domains of a layer of masked Boolean functions. 122

9.1 Overview interaction active adversary. 132
9.2 Propagation of faults example. 133
9.3 The effect of an area-extended fault. 145
9.4 The effect of a permanent fault. 146
9.5 A combined secure tiled multiplier. 148

List of Tables

3.1 Cost of random number generators using cryptographic primitives. 34
3.2 Cost of maskings including randomness. 35
3.3 Cost of LFSRs. 36

4.1 Cost of our first-order AES including the cost for randomness. . 49

6.1 Share dependencies in the second-order AES. 90

10.1 Activity patterns between two S-boxes. 156

Chapter 1

Introduction

Ever since people could write there was a need for cryptography. From the Roman Empire encoding messages to their battalions to the seemingly impossible to break Enigma employed by Germany during the Second World War. With the rise of information technology, cryptography drastically changed. It became automated, running in the background of every device around. Currently, people make use of some cryptographic protocol every time they browse the web, pay with a card, or open their phone.

The Rise of Symmetric Cryptography One of the main building blocks that lead to this advancement is symmetric-key cryptography. The idea behind symmetric cryptography is the following.

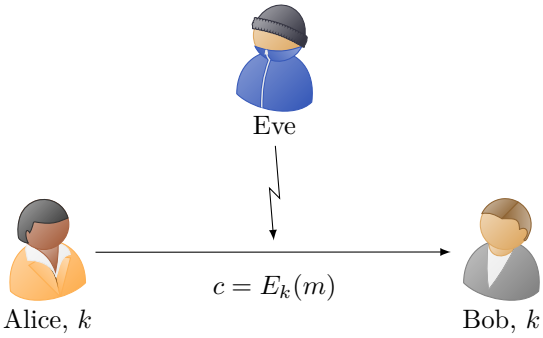


Figure 1.1: Alice and Bob communicating via a symmetric-key cipher.

Alice and Bob both share a secret key, a number only the two of them know. Using this key, the two can communicate without anyone else being able to understand them. This works as follows. If Alice wants to encrypt a message m and send it to Bob, she uses the secret key k and an algorithm, or *cipher*, E to transform the message to $c = E_k(m)$. This message is then sent to Bob who can decrypt it by using k and the inverse calculation $E_k^{-1}(c) = m$. The cipher E has the property that an eavesdropper, Eve, can not figure out m or k from the ciphertext c even if Eve knows the algorithm E. That is, if Eve only has limited time and computational power.

Ciphers typically are built from *block ciphers* or *cryptographic permutations* using some mode of operation. That is, building blocks handle short messages while modes of operation handle processing long messages or multiple messages. In the early 1970s a revolution in the world of cryptography happened. At IBM, a man named Horst Feistel was designing a mathematical algorithm called Lucifer. This algorithm, after modifications, later became known as the *Data Encryption Standard* or DES for short. This was a standardised block cipher which everyone was allowed to use to secure information technology around the world. Nevertheless, the original design by Feistel was changed such that it was weakened to brute force attacks (and in fairness also strengthened against differential attacks) such that the National Security Agency (NSA), and thus also everyone else, could break the encryption when needed. In order to prevent this break in security and privacy, in 2001 a new encryption standard, the *Advanced Encryption Standard* or AES, was made. The design was chosen from a public competition, where two Belgians by the names of Joan Daemen and Vincent Rijmen won. Over more than twenty years, the AES has since been the first line of defence against cyber crime.

The Threat against Embedded Devices Twenty years of academic research and industrial verification were not able to make a dent in the AES's mathematical security. And yet, just before the AES winner was chosen, a new article became public to the world. In 1999, Kocher, Jaffe, and Jun [73] published an article that would create an entirely new branch of cryptography called *side-channel analysis*.

Embedded devices such as smart cards or smartphones have become legion. We carry cards to open the doors in our office, bank cards or credit cards to pay in stores, even our identity cards carry a chip with our personal data and a key to sign documents with. The security of the devices remained the same, using AES, but the attacker has changed. This new attacker doesn't just eavesdrop on the communication sent from your cards. They can steal the device itself. Perhaps the attacker is even the owner of the card. Think about owning a

metro card and wanting to top it up without paying.

With an attacker holding the device itself, the threat model has changed drastically. The attacker can tamper with the device and read secret information on it. But how do you know what is happening inside such a tiny chip on a plastic card? This is where side-channel analysis comes in.

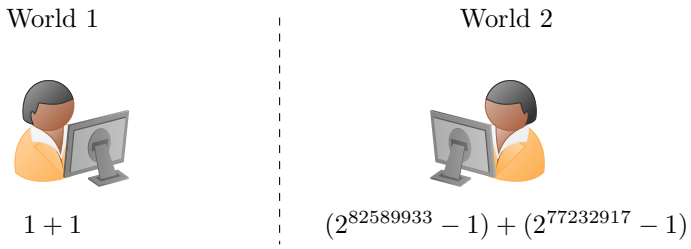


Figure 1.2: Alice performing one of two operations on a computer.

Consider Alice has her computer and chooses to do one of two calculations. Either she will add 1 and 1 together, or she adds the two largest known prime numbers. Would it be possible to know which of the two operations she made without asking her or looking at her screen?

If we knew when the computation was finished, we could figure out which of the two operations was performed. Our computers take time to perform calculations, thus the world where $1 + 1$ is performed will take much less time than the second world. Similarly, if we knew how much electricity the computer needed we could also better guess which operation was performed. Simply put, the computer had to work much harder to find the answer to the second world's sum as opposed to the first world. Working harder means that the computer starts to heat up more, requires more power, and makes more noise. Any of these physical traits could be used to figure out in which world Alice made the computational request. In the thesis, we focus on side-channel leakage originating from the power consumption of a device.

Sure, it is easy to see that there is a difference in power consumption between a sum like $1 + 1$ and summing two of the largest known prime numbers. But the remarkable aspect of side-channel analysis lies in its use of engineering and statistics. Consider the above experiment again, but now in the first world Alice is asked to compute $1+0$ and in the second $1+1$. From the field of hardware, take for example Complementary Metal Oxide Semiconductor (CMOS) technology, we learn that there is a different power consumption between the two operations. Consider that we can ask Alice to perform the same calculation thousands of

times. By averaging the power consumption over the thousands of repetitions, we can remove noise from the measuring equipment or noise inherent to the device. After enough data is gathered, we can find out in which world Alice is, no matter which two operations she chooses. The same is applied on a device such as a smart card to figure out which data is flowing through it.

The attack by Kocher *et al.* in 1999, called *Differential Power Analysis* or DPA, uses the above insight to attack block ciphers such as DES and AES. The strength of this attack lies in its efficacy. Typically, the attacker only needs to measure the device a thousand times to retrieve the secret key of the cipher. Compare this to requiring 2^{128} queries to break the AES using a brute force approach. DPA typically breaks a device requiring mere minutes, meaning that this simple attack suddenly threatens each physical device the attacker could get a hold off.

Countering Side-Channel Analysis As quick as side-channel analysis appeared, countermeasures also started to appear. Hardware engineers immediately tried to balance the power consumption independent of the input of the device using their knowledge of the underlying technology. Security experts grabbed after obfuscation methods where dummy operations are performed or random delays were inserted. However, one algorithmic method grabbed the attention both of academics and industry. This countermeasure, called *masking*, was independently introduced in 1999 by Goubin and Patarin [61] and Chari *et al.* [30]. The idea is to split the computation in multiple parts such that attackers, only observing some of the parts, gain no information on the whole. This forces attackers to observe the simultaneous power consumption of each of the parts. Therein lies the key to the countermeasure. The measurements of the power consumption has inherent noise and this noise can be significant in practice. Masking achieves what is called *noise amplification*. Due to an attacker needing to observe the power consumption of multiple parts at the same time, the noise on the combined data is exponentially increased. Whereas a side-channel attack on an unprotected AES requires around a thousand data points, or *traces*, a two-part masked AES would require a million traces to break.

Apart from the improved security of masking methods as outlined above, they have captured the attention of industry because of their flexibility in the cost. One can make maskings such that they minimise the latency or the area cost. That means the industry can easily match this method to their solutions. The method also grabbed the interest of academia as security models could be made to quantify (if rather crudely) their security. As a result, academia and industry have been able to work closely together over the last twenty years to improve

the method.

A Solution at a Cost Here comes the twist. Masking methods have been theoretically and practically shown to provide tangible security gains time and time again. Nevertheless, masking does come at a cost. This cost includes an increase in area, latency, and power or energy consumption. And this increase in costs is not trivial. Which has resulted in the cryptographic community working to lower them as much as possible. However, these optimisations were not made without flaws. Masking schemes have often been broken or shown to not adhere to their security claims. It is from the rise of security models that masking schemes have been increasingly resilient over the past years. Nevertheless, these security models only cover so much of the practical attacker's capabilities.

A current assumption that has often been overlooked in our models is the integration of a random number generator. These generators are divided in two groups, non-deterministic random number generators and pseudo-random number generators. While non-deterministic random number generators sample from an entropy pool created by physical sources such as variations in heat, pseudo-random number generators are built on cryptographic primitives such as the AES. Often security models for side-channel countermeasures assume that randomness is freely available. However, the generation of this randomness is often expensive and insecurely implemented in practice. A plug-and-play integration of such a generator can also lead to weaknesses in the system. Our security models still have room to grow.

An Even Older Threat Even before the work by Kocher *et al.*, in 1997 Biham and Shamir [13] discussed the exploitation of a fault injected into a block cipher. They describe an easy-to-use technique to break ciphers in under a dozen faulty ciphertexts. While the injection of a precise fault in an embedded device nears art more than science, the techniques have been rapidly advancing. However, the cryptographic community has not responded as quickly as with side-channel analysis. While some countermeasures are known, none have been extensively investigated. This can be explained from the inherent cost of lab experiments as devices are easily destroyed using fault injection techniques. But even then, the theory has also been lagging behind.

Currently, the seemingly standard countermeasure is to encode the algorithm using error correcting codes. The most popular among them is duplication, where the algorithm is implemented twice is order to verify whether both parts output the same result. However, no security models are currently known to investigate the security of such an encoding. The works discussing the security of encoded designs is sparse and often follow completely different security analyses.

From the few countermeasures that have been proposed, several have already been broken repeating the pattern we saw with masking schemes. Lessons should be learned from the research in side-channel analysis and be applied to the field of fault analysis. Security models that relate to practice and still allow for efficient countermeasures should be proposed. From such a model, the theory of fault countermeasures can more easily grow.

1.1 The Context of the Thesis

We sketch the context of the work and name the assumptions on which the work is built.

Masking In this thesis, we assume that masking is the best solution to counter side-channel analysis. While the efficacy of masking has been proven time and time again in hardware, it is impossible to prove it is the most efficient countermeasure. Apart from the theoretical question, we see that other methods such as hiding or leakage resilience exist and also have shown promising results. It remains to be seen whether one of those methods will outweigh the other or whether a solution is found in a combination of them. Nevertheless, masking is currently a very popular countermeasure both in academia and in industry.

Hardware We aim to protect hardware implementations such as designs on Field Programmable Gate Arrays (FPGA) or Application Specific Integrated Circuits (ASIC). This excludes maskings on software such as micro-controllers, smartphones, or desktops. Thus, we take specific care to protect against effects which occur on hardware such as glitches and Hamming distance leakage specific to CMOS technology.

Recent research showed that masking schemes in software are not evidently secure and thus the security claims from the theory deviate from practice. More specifically, at CHES 2020 during a public competition, it was shown that even sixth-order masked schemes could be broken using as few as 4000 traces. This shows that the field of software masking is in dire need for radical changes which are outside the scope of this thesis. Instead, the same competition at CHES 2020 ended with their second-order hardware implementation to be unbroken. This shows the validity of masking as a countermeasure on hardware.

The Probing Model Several security models exist in the field of side-channel analysis. However, the threshold probing model has been growing out to be the

standard security model in the field. Nevertheless, the model is far from perfect. It is known to be both too strong and too weak. The model does not include measurement noise which is the corner stone of side-channel analysis making the model too strong. At the same time, it does not include effects that occur in practice such as glitches, transitions, or couplings, or attacks which are not captured by the model such as horizontal attacks making the model too weak. In this thesis, we build countermeasures which are secure in the probing security model. We assume this model allows designers to create countermeasures which provide practical security.

A Source of Randomness In this thesis, we investigate the reduction of randomness of masking schemes. However, most of the times a masking still needs some random bits to refresh the master key or to share the plaintext. That means we have to assume some random bits can be created by the design. In other words, we assume there is either a good source of entropy available or a secure pseudo-random number generator from which the output can be used to instantiate the masking.

Active Security Concerning active security, not much research has been done. The practice lies far away from the theory. To that end, we assume a basic fault adversary which can only target a threshold number of resources in the platform. While this might not fully capture practice, it allows for a preliminary theoretical study of the field. We hope that over time these adversaries are extended to better approximate reality.

1.2 Research Directions

The research is divided in two main paths. The first tackles side-channel analysis where the aim is to analyse the randomness usage in masking schemes with the goal to improve its efficiency and to be assured of its security. The second contributes security models in the field of active and combined security with the goal to provide some preliminary countermeasures against these attacks. Each chapter in the thesis can be seen as the posing and answering of a more specific research question.

Randomness Usage in Maskings The first path in the thesis tackles the randomness usage of masking schemes. We find that the randomness cost reported in academic articles is often neglected and methods to generate it

are not given. There is a need to investigate the cost of this generation. As the cryptographic secure generation of randomness is often too expensive, non-cryptographic techniques are used instead. The security of using non-cryptographic techniques has not yet been challenged by the cryptographic community, although a general distrust towards them exists. Instead, in this thesis, we investigate how to reduce the randomness cost of a masking up to the point it requires only a small initial injection of randomness.

We investigate *threshold implementations* as a starting point of the research. It is known that certain threshold implementations achieve security without using fresh randomness. However, these techniques are not always easy to apply to ciphers like the AES, or their cost, such as area or latency, is higher than their randomness-using counterparts. Furthermore, threshold implementations only provide protection against first-order attacks. Ensuring higher-order security seems to inevitably require fresh randomness.

The path of this research direction concerns itself with the security analysis of masking schemes including their randomness cost and the generalisation of the properties of threshold implementations to ensure higher-order security without losing their unique functionality.

Active and Combined Security Models The second path of the thesis handles active- and combined-security models. Not much research has been done on algorithmic countermeasures against fault attacks, let alone combined attacks where the attacker uses both fault attacks and side-channel analysis or when an adversary uses a fault to receive side-channel information. It is not yet well understood what an adversary's capabilities would be and even though some countermeasures are being proposed, most seem to contain mistakes due to the lack of tools or formal security models.

Thus, the second research path handles the proposition of algorithmic adversary models to capture fault or combined attackers. Security models around these adversaries should be created to assure designers can easily make or verify countermeasures. From the success of the non-interference model for passive security, it is clear that the notion of composable security helps to provide the first steps in the field. From such a composable security notion, one only needs to create a secure multiplier and adder to secure arbitrary computation.

Recent work also uses multiparty computational methods to secure against faults. While this is a good first step, the theory behind it is lacking. The multiparty security model is not related to other existing models and it is not clear how to create countermeasures which are efficient in hardware. Neither is it clear how their adversary relates to practice. Thus, we aim to bring this

multiparty computation model closer to the field of side-channel and fault analysis.

1.3 Road Map of the Thesis

This thesis consists of ten chapters. We go over each chapter, give a brief description of it, and highlight our own contributions and published works.

Chapter 1 The first chapter, *i.e.* this chapter, concerns the introduction to the thesis. We provide the context to symmetric key cryptography and outline the danger of power analysis to implementations on embedded devices. We introduce masking as the centre countermeasure. We discuss its cost and its security in light of its randomness usage. We also discuss fault attacks and the current lack of theory in the field. Those problems form the two main research directions of the thesis.

Chapter 2 The second chapter introduces masking. We provide a brief introduction to hardware, provide the definition of Boolean masking, and outline its practical security. We then introduce the probing adversary, which has become the standard adversary in the field of side-channel analysis, and we discuss its extensions to capture glitch and transition effects. After the adversary model, the security model is discussed. We consider three models, the original probing security model, the non-interference composable security framework, and our own bounded-query model. The chapter finishes by providing a first-order masked XOR and AND gate for which the security is proven in the non-interference model.

The writing of this chapter is based on previous known works.

Chapter 3 The third chapter discusses the need and cost of randomness in maskings. We discuss that composable security models need randomness. We investigate the use of probing robust random generators. We introduce non-deterministic random number generators and random number generators using cryptographic primitives. We give an overview of the cost of those cryptographic generators to create a random bit per cycle. We apply those costs to the two most known maskings of the AES. We then investigate linear feedback shift registers as a lightweight option to generate the randomness in a masking. We provide three theoretic attacks to show that these linear systems can make the masking insecure.

The writing of this chapter is based on previous known works.

Chapter 4 The fourth chapter introduces threshold implementations. These are maskings which comply to two properties: non-completeness and uniformity. We provide the definitions related to threshold implementations and prove their security in the probing model. We show that threshold implementations can differ from non-interferent secure solutions and that certain threshold implementations secure their operations without requiring fresh randomness. We then discuss the need for uniformity in maskings. Finally, we generalise a version of the changing of the guards technique and apply it to create a first-order masking of the AES which requires no fresh randomness.

This chapter is based on work published at the workshop of Theory of Implementation Security at CCS 2019 [52] and was supervised by Svetla Nikova and Vincent Rijmen.

Chapter 5 The fifth chapter discusses higher-order threshold implementations. We provide some preliminaries on Fourier analysis and discuss a reduction from entropy to the Fourier transform of the distribution. This is used to provide an upper bound on a probing adversary. We show how linear cryptanalysis can be used on masked variables allowing for a trail-wise analysis of maskings. In particular, for a multi-round analysis. The single round analysis is based on typical side-channel methods and uses the property of higher-order non-completeness. We showcase the analysis on a masking of the LED cipher. This cipher has no need for fresh randomness to secure its operation and still attains second-order probing security in a bounded-query model.

This chapter is based on joint work with Tim Beyne and Zhenda Zhang and is published at Asiacrypt 2020 [12] where I am a main author.

Chapter 6 The sixth chapter discusses techniques to improve the efficiency of higher-order threshold implementations. We introduce a method to relax the second-order non-completeness requirement for maskings which reduces the number of needed shares and a method to remove register stages to improve the latency. We provide an efficient sharing of LED using the minimal number of shares and with low latency. We also provide a second variant which generates as much randomness as it requires. We improve the diffusion of the changing of the guards method such that it can be used in a second-order context and apply this to make a second-order sharing of the AES. Finally, we improve the glitch model by sharpening the theory and using practical validation to provide a better bound for the adversary.

This chapter is based on joint work with Tim Beyne, Adrián Ranea, Danilo Šijačić, Aein Rezaei Shahmirzadi, and Amir Moradi where one part is accepted at Selected Areas in Cryptography (SAC) 2021, the preprint can be found online [11], and the other part is accepted at CHES 2022, which can also be found online [10]. In both works I am a main author.

Chapter 7 The seventh chapter describes the notion of resilient uniformity. This is a generalisation of the uniformity notion to allow for higher-order security. We provide the definitions of the generalised uniformity and non-completeness notions, and prove their security against a second-order probing adversary. We then link the notion to the "consolidating masking schemes" work to formally prove this is secure.

This chapter is based on my work published at the Cryptography and Communications journal which was supervised by Svetla Nikova and can be found on ePrint [51].

Chapter 8 The eight chapter uses the analysis method from Chapter 5 to analyse the security of a masking against more advanced leakage effects. We recall transition effects and discuss the security of a PRESENT, AES, and ASCON masking. The security analysis considers a black-box S-box and instead uses the diffusion properties of the cipher to leverage costs of the masking. Afterwards, coupling effects are described and the combined effect of glitches, transitions, and couplings are investigated. We show that one can easily protect a masking against these advanced leakage effects by using the diffusion properties of the primitive.

This chapter is based on work accepted at Indocrypt 2021 and can be found on ePrint [48].

Chapter 9 The ninth chapter concerns the second research direction of the thesis. We start by quickly introducing fault injection mechanisms and the cryptographic assumptions of embedded implementations. We then give the definition of encoding methods with a focus on duplication of shares. We then introduce the active and combined adversary together with a simulation based security model. We provide the NINA composable combined security model and provide a secure AND gate. The security of this gate is formally proven in the NINA model. We then extend the active and combined adversaries by extending their fault capabilities. We discuss the tile model and the tile adversary. We link this tile adversary with the fault extended adversary and discuss composable

security. Finally, we provide a tiled multiplier which we show is secure against an advanced fault or probe.

This chapter is based on work where one part is published at CT-RSA 2020 [50] and the other part at CARDIS 2020 [49]. In both works I am the main author and was supervised by Svetla Nikova.

Chapter 10 The last chapter concerns the conclusion and future work of the thesis. We provide a summary of the entire thesis and conclude the research directions proposed in the first chapter. We conclude the work by providing several open problems and interesting future research directions.

Chapter 2

Masking as a Countermeasure

In this chapter, we discuss the object of study for the thesis, namely masking. This algorithmic technique is used as a countermeasure against side-channel attacks. We discuss the motivation of masking, give its definition, and discuss the probing adversary with its security models. We end by giving examples of a masked addition and multiplication operation.

Section 2.6.3 is written by myself and contains unpublished novel ideas. The rest of this chapter is written by myself but is based on previous works.

2.1 Motivation

Previously, we discussed side-channel attacks and saw their potential danger for the security of embedded symmetric primitives. In the literature that followed the work by Kocher *et al.* [73] several countermeasures were proposed. These spanned from trying to balance the power consumption of the device independent of the data, see for example *dual rail logic* [79], creating additional noise for the attacker by shuffling operations, or by creating random delays, see the comprehensive study by Veyrat-Charvillon *et al.* [99]. These previous examples are often called hiding techniques. Some solutions on the mode of operation were also devised, so-called *leakage resilient modes*, a survey on leakage resilience is made by Kalai *et al.* [70]. The most intuitive methods come down to cryptographically re-keying the primitive such that the attacker only has limited data on a single key. Seeing that this data is noisy, one can easily deter attacks.

The previous solutions are useful. However, hiding techniques typically only offer a linear increase in the noise. That means that hiding methods alone will not prove sufficient to protect against practical attacks. In contrast, provable security methods like leakage resiliency remain quite untested in practice and their security proofs in an information-theoretic setting create complicated or vague claims which can be quite far from practice. Instead, there is a third solution which resides in the space of algorithmic countermeasures called *masking*. A masked version of a symmetric primitive can be seen as splitting the primitive in multiple parts. Each part separately seems to work only on random data. However, when combining the parts one can reconstruct the ciphertext. Intuitively, to attack a masked countermeasure the attacker needs to observe separate parts of the computation at the same time. This makes it much more difficult for the attacker to retrieve secret data. The attacker would require a significant increased number of traces for the attack to succeed.

Masking has become a popular countermeasure both in academia and in industry. We can create security models which approximate reality and prove masking provides a polynomial increase in the noise making them stand out between other methods like hiding. Masking is also easy to implement and many trade-offs between costs like area or latency are possible. They also have been tested over the past twenty years and, in hardware, masking methods undoubtedly increase the practical security of implementations. It is because of both the theoretic as practical merits that masking is the object of study in this thesis for countermeasures against side-channel analysis.

2.2 Introduction to Hardware

Before giving the definition of masking, we repeat that the thesis handles maskings specifically on hardware. As such, we first recall the basics of hardware.

Synchronous Circuits A synchronous circuit consists of combinatorial gates (AND, XOR, etc.) and sequential logic (memory, registers). When the circuit is powered on, all registers, gates, and wires are powered too at which point they all carry a digital value. There is a clock synchronising the operations of different circuit elements. A clock cycle is the time between two clock ticks. In each clock cycle the combinatorial logic is re-evaluated and results are stored in the registers.

Registers A register (or memory cell) has one input and one output and its functionality is controlled by the clock. Registers release a signal by opening its

out-line while the input-line is closed (only one is open at a time). The register out-line stays open until the signal in the logic becomes stable, after that it stores the newly computed value - hence the register closes the output-line and opens the input-line. This is indicated in Figure 2.1.

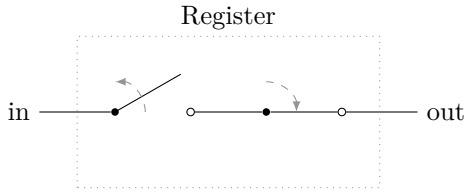


Figure 2.1: Abstract depiction of a register.

Logical Gates Logical gates perform simple Boolean operations. They have several wires as input and a single wire as output. Each gate can have a different time to propagate a signal from its inputs to its output and each gate can have a different power consumption. A change of its inputs causes re-evaluation of the gate and hence may change the output value.

2.3 Boolean Masking

Masking methods are essentially secret sharing methods. The difference with secret sharing lies in how computation on shares is performed. Since the adversary of typical secret sharing methods, a computation between several parties where some are malicious, is different from the side-channel adversary the solutions on how to operate on shared/masked data are often very different.

We start by defining Boolean masking. This method was independently introduced by Goubin and Patarin [61] and Chari *et al.* [30].

Definition 2.1 (Boolean masking [30,61])**.** The s-shared Boolean masking of a variable $x \in \mathbb{F}_2^n$ consists of a vector $(x^1, \ldots, x^s) \in (\mathbb{F}_2^n)^s$ such that $x = \sum_{i=1}^{s} x^i$.

In a physical countermeasure, such a vector is made by using random bits. For example, to share a secret x in two shares one can use a random bit r and create the vector $(x + r, r) = (x^1, x^2)$. That way, each share x^1 or x^2 is uniform random. Later in Chapter 4, we define a uniform masking to reflect this property.

Other sharing methods also exist. For example, there is Shamir's secret sharing using Reed-Solomon codes [95] which is also used in physical countermeasures

often going by the name *polynomial masking* [60]. Other examples are found as *multiplicative maskings*, an example on AES is made by De Meyer *et al.* [47], or *inner-product maskings*, an example is found in the work by Balasch *et al.* [2]. In this thesis, we will mainly consider countermeasures using Boolean masking as these schemes are typically the most lightweight for implementations.

2.4 Intuition on Security

In this section, we provide some easy-to-understand argument why masking, and more specifically Boolean masking, provides an improved protection against side-channel attacks.

We consider the adversary and security model proposed by Barthe *et al.* [5] called the *bounded moment leakage model*. This model is quite close to practice as attackers typically use raw probabilistic moments to find leakage. We start by defining mixed moments.

Definition 2.2 (Mixed moment at orders $o_1, o_2, ..., o_r$ [5])**.** Let $\{X_i\}_{i=1}^r$ be a set of r random variables. The mixed moment at orders $o_1, ..., o_r$ of $\{X_i\}_{i=1}^r$ is:

$$\mathbb{E}(X_1^{o_1} \times X_2^{o_2} \times ... \times X_r^{o_r})$$

where $\mathbb{E}$ denotes the expectation operator and $\times$ denotes the multiplication in $\mathbb{R}$. For simplicity, we denote the integer $o = \sum_i o_i$ as the order of this mixed moment. We further say that a mixed moment at order o is m-variate (or has dimension m) if there are exactly m non-zero coefficients o_i.

The above easily explains the difference between univariate and multivariate attacks. A *univariate* attack is one where the trace is preprocessed using a high-degree function and afterwards the straightforward DPA attack is performed. This relates to the calculation of a higher-order moment $E(X^d)$ for $d > 1$. A *multivariate* attack is one where multiple points on the trace are combined and the DPA attack is performed on the combination of points. This relates to the calculation of a mixed moment $E(X_1 \times ...X_d)$ for $d > 1$. Of course, the two attack methods can be combined giving the general definition above.

We then define the bounded moment security model.

Definition 2.3 (Security in the bounded moment model [5])**.** Let $\{L_c\}_{c=1}^N$ be the leakage vectors corresponding to an N-cycle cryptographic implementation manipulating a secret variable Y. This implementation is secure at order o if all the mixed moments of order up to o of $\{L_c\}_{c=1}^N$ are independent of Y.

Consider an implementation where a share vector $\bar{x} = (x^1, ..., x^s)$ is loaded into memory, one share per cycle. Consider an attacker taking $d \leq s - 1$ samples $\{L_i\}_{i=1}^d$ from the trace of this computation. The higher-order variant of DPA tries to distinguish the moment $\mathbb{E}(L^1, ..., L^d)$ for different secrets x. However, since any d shares from $\bar{x}$ are jointly uniform random, it is clear that this attack must fail. As a result, in order to have a successful attack one must take at least a mixed moment of order s.

The work by Barthe *et al.* [5] showed that the above argument also holds when parallel processing the same computation. Meaning that the implementation loading in $\bar{x}$ as a whole in one cycle is also secure in the bounded moment model as long as the mixed moments are of order $d \leq s - 1$. As a result, in order to break either the serial or parallel variant of the above implementation, the attacker should use moments of order s or higher of the data provided in the traces. The assumption is that estimating higher-order moments is significantly more difficult and will require more traces. The idea behind this is called *noise amplification*. Calculating higher-order mixed moments provide a new stochastic variable on which the noise is higher. This noise amplification is dependent on which leakage function is taken. In hardware, a popular leakage function is the Hamming weight function with additive Gaussian noise. Meaning $L(X) = HW(X) + \mathcal{N}(0, \sigma)$. In case a mixed moment of order with $\sum_i o_i = d$ is chosen, we find that

$$L(X_1^{o_1} \times X_2^{o_2} \times ... \times X_r^{o_r}) \sim HW(X_1^{o_1}) \times HW(X_2^{o_2}) \times ... \times HW(X_r^{o_r}) + \mathcal{N}(0, \sigma^d).$$

It is the noise amplification property which lies at the base of masking. In practice, the attacker can typically only gather less than a billion traces and estimating its mean typically requires around a thousand data points. That means that in practice $s = 2$ or $s = 3$ typically suffices. Moreover, industry typically pairs masking with hiding techniques to increase the noise and thus further increase the number of data points required to estimate its mean.

While the above implementation was quite simple only loading the shares into memory, the security of more complicated calculations will rely on what is called the *order of security*. A dth-order secure scheme, more formally defined in Section 2.6, is one where the adversary would need to combine at least $d + 1$ points in the trace or use at least a $d + 1$ order moment in order to break the scheme. This order of security can deviate from the number of Boolean shares. Take for example the masked AND gate depicted in Figure 2.2. The mean

$$E(HW(c^1) + \mathcal{N}(0, \sigma)) = c^1 = a^1 b^1 \oplus a^1 b^2 = a^1 b$$

is different depending on the secret b and, as a result, the operation is insecure against a first-order moment even though the operation is shared in two.

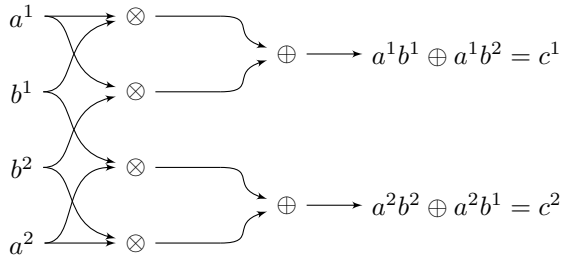


Figure 2.2: Depiction of an insecure first-order masked AND.

2.5 The Probing Adversary Model

The previous implementation captured the basic idea of masking. However, in practice more complicated implementations are used. These implementations are typically prone to errors and verifying them can be complex. The above security model, the bounded moment model, is difficult to use in practice. Instead typically a different model is utilised, namely the *probing model*. We note that by probe, we do not mean a physical probe such as an EM probe. Instead, the word probe is used as an abstract concept through which an adversary can perfectly observe a part of the computation. This model was originally proposed by Ishai *et al.* [69] in 2003. Even though the model is quite abstract and seemingly far from reality, it has enjoyed a lot of attention from both the theory side of the community in the form of reductions to and from other security models and the practice side of academia in the form of the practical verification of probing secure designs.

2.5.1 The Circuit Model

Before we start with the probing model, we need some agreement on how to represent the computation. For this reason, we introduce the circuit model. We represent computations in arithmetic circuit form, a directed acyclic graph whose nodes are operations over a finite field $\mathbb{F}$ and whose edges are wires. Additionally, we consider probabilistic arithmetic circuits, meaning circuits with nodes having no input and uniform random elements over $\mathbb{F}$ as output; this randomness is independent and identically distributed, and the correctness of the circuit is not dependent on it.

In short, we write down the hardware implementation as a rolled-out circuit consisting of operations over the finite field $\mathbb{F}$, typically over $\mathbb{F}_2$ as XORs and

ANDs. Essentially, a netlist with a limited number of allowed gates.

2.5.2 The Probing Adversary Model

The probing model consists of a d-threshold-probing adversary which can observe at most d wires of the circuit in a single query (single-shot model). By default, the adversary is allowed to adaptively move its probes between queries, but not within a query. The adversary cannot attack registers and gates, and cannot observe the several values flowing through a wire in a clock cycle caused by glitches.

More formally, let $\ell \geq t$ be positive integers. A *t-threshold-probing adversary on $\mathbb{F}_2^\ell$* is an algorithm $\mathcal{A}$ that interacts as follows with an oracle (the masked primitive) that holds an arbitrary sequence $(x_1, ..., x_\ell) \in \mathbb{F}_2^\ell$:

1. $\mathcal{A}$ may specify a set $\mathcal{I} = \{i_1, ..., i_{|\mathcal{I}|}\} \subset \{1, ..., \ell\}$ of cardinality at most t,
2. $\mathcal{A}$ then receives $(x_{i_1}, \ldots, x_{i_{|\mathcal{I}|}})$.

Note in particular that the adversary $\mathcal{A}$ is computationally unbounded.

The original definition given by Ishai *et al.* is different from the above definition which has become the standard in the field. In the original definition, the adversary was allowed to probe d wires each clock cycle. This original model is now called the *continuous probing model* which is still researched but rather as a model to capture more advanced leakage effects.

Barthe *et al.* [5] showed that a circuit which is d^{th}-order secure in the probing model is also d^{th}-order secure in the bounded moment model. This shows that even though the probing model looks abstract and far away from the realistic capabilities of the attacker, it does capture the attacker up to some extent. On top of that, it is very easy to determine the security of the circuit in the probing model. Essentially, it is the probing model which allowed designers to make provably secure masked circuits.

Even though the probing model captures some of the adversarial effects, there are several physical effects which are not captured. We go through two of the most important ones, glitches and transition effects.

2.5.3 Modelling Glitches

The first physical effect we discuss is a *glitch*. This effect causes maskings on hardware to look different than those on software as glitches mainly have their

effect in hardware.

Let us consider basic combinatorial logic, namely the logic which connects two layers of registers. When a cycle starts, the output signals of the registers are inputs for the basic logic and these signals will start propagating through the wires and the gates until they reach the output registers. Gate evaluation may happen several times until the signals (and hence the gate) become stable. We go through three reasons why this might happen: a) the wire signals propagate with different speed; b) the wires have different length; and c) each gate has different propagation time. We will refer to these value changes on the wires and gates as glitches.

In a cycle there are two main phases. The first phase is one in which the wires and gates do not have a stable value. This phase is followed by one in which all values are stable. The power consumption at a time sample is the sum of the power consumption of the wires, gates, and registers belonging to this simple logic. For CMOS technologies, the power consumption during the first phase is higher and more apt to change because of glitches compared to the second phase. We stress that glitches occur in the logic between two memory gates and are stopped by registers.

Fact: Glitches do not propagate through memory gates.

Glitches can result in significant leakage that is not accounted for by the standard probing model, see for example the attacks of Mangard *et al.* on several masked AES implementations [75]. Consequently, it is necessary to extend the capabilities of threshold probing adversaries in order to capture the physical effect of glitches on a hardware platform. Whereas a probe normally results in the value of a single wire, a glitch-extended probe allows observing all value used in the calculation of the probed wire up to the previous register layer. This extension of the probing model has been discussed by several authors, here we give the definition from Faust *et al.* [59] who describes it as follows:

"**Specific model for glitches.** For any ϵ-input circuit G, combinatorial recombinations (aka glitches) can be modelled with specifically ϵ-extended probes so that probing any output of the function allows the adversary to observe all its ϵ inputs."

2.5.4 Transition Leakage

The second effect we discuss are transitions. This effect occurs both in hardware as in software.

Consider registers as recalled in Section 2.2. When the register input is open to store the incoming value, the new value has to overwrite the so far stored value. If these are different values then the attacker can measure a peak in the power consumption compared to the case when the values are the same – this is called transition leakage. Similar to registers, if a new value different from the wire current value starts propagating through a wire then the power consumption will differ compared to the case when the new and the current value are the same.

As a result, if in a memory cell the element x is erased and instead y is stored, transitions can leak both x and y. We integrate such leakage effects in the probing model following the work by Faust *et al.* [59]. There the model is described as follows.

"**Specific model for transitions.** For a memory cell m, memory recombinations (aka transitions) can be modelled with specifically 2-extended probes so that probing m allows the adversary to observe any pair of values stored in 2 of its consecutive invocations."

As mentioned above, transition leakage does not only occur in memory elements. Thus, we extend the description such that the extended leakage is viewed in any gate or wire.

The work by Ishai *et al.* [69] and the publications that followed considered a circuit model that represents a deterministic circuit as a directed acyclic graph whose vertices are combinatorial gates and its edges are wires carrying elements from a finite field. However such a simplification of the circuit model does not take into account the circuit topology. While the leakage of glitches does not depend on the circuit architecture/topology, for transitions the leakage is mainly influenced by the circuit's architecture. As a result, to study transition effects one should study circuits with loops and a notion of time and consider particular architectures.

2.6 Security Models for the Probing Model

In the previous section, we discussed the current standard adversary model used to model side-channel attacks. In this section, we discuss the non-interference framework which provides composable security for the probing model and the bounded-query model which we show later on allows for low-randomness maskings.

2.6.1 The Probing Security Model

The first security game we consider is the original one by Ishai *et al.* [69]. For ease, this model is simply called the *probing security model*. In particular, perfect security is considered where the adversaries are computationally unbounded and must specify probe locations before querying the circuit. Take an arbitrary d-threshold-probing adversary $\mathcal{A}$ and $\mathsf{Dec} \circ C' \circ \mathsf{Enc}$ a randomised stateless sequence (circuit) which is the masked version of C. Importantly, the adversary's interaction with the circuit is mediated through the encoder and decoder algorithms Enc and Dec, neither of which can be probed since both functions mask or unmask the secret information.

There are two notions of security for stateless circuits. The circuit C' is considered *sound* if for all inputs k,

$$\mathsf{Dec}(C'(\mathsf{Enc}(k))) = C(k).$$

The circuit C' is d^{th}-order *private* if it can be simulated from scratch, meaning the simulator is given nothing, such that no d-threshold-probing adversary $\mathcal{A}$ can distinguish $\mathsf{Dec} \circ C' \circ \mathsf{Enc}$ from the simulation.

A circuit is considered d^{th}-order probing secure if it is both sound and d^{th}-order private following the above games.

2.6.2 The Non-Interference Framework

Typically it is computationally infeasible to prove the first or higher-order probing security of a large circuit. Instead, one proves the composable security of several smaller circuits of which the larger one is composed. But how do we know that the composition of those smaller circuits is again secure? In fact it was illustrated by Coron *et al.* [33] that this is not always the case. To assure this, we need a stronger security model which provides *composable security*. The composable security notion for probing security has been studied by Barthe et al. [4]. The framework is built on simulation-based security arguments.

We introduce the simulation framework for probing security. Here the adversary (distinguisher) is either interacting with the actual circuit or with a simulator. This simulator is given only a part of the input and does not know the secrets of the circuit. The distinguisher's goal is to determine whether it is interacting with the simulator or with the actual circuit. A failure to do so implies that the adversary can know at most the shares given to the simulator and as a result only some inputs of the circuit.

Definition 2.4 (Simulatability [7])**.** Let $P = \{p_1, ..., p_d\}$ be a set of d probes of a circuit C with m inputs where each input is divided into n shares. Let the set of q shares of each input given to the simulator be denoted by $I = \{(i_1, j_1), ..., (i_m, j_q)\} \subset \{1, ..., m\} \times \{1, ..., n\}$. Denote C_P as the circuit C with probed wires as per P.

We define the simulator S and distinguisher D as the following probabilistic functions.

$$S : \mathbb{F}^q \times \mathbb{F}^m \to \mathbb{F}^d$$

$$D : \mathbb{F}^d \times \mathbb{F}^{nm} \to \{0, 1\}$$

We say that the set of probes P of the circuit C can be simulated with the set of values on the input wires I if there exists a simulator S, such that for any distinguisher D and any inputs $a_{*,*}$, we have that

$$\left| \Pr[D(C_P(a^{*,*}), a^{*,*}) = 1] - \Pr[D(S(I), a^{*,*}) = 1] \right| = 0$$

where the probability is taken over the random coins in C, S and D.

The above notion of simulation is then used to provide the first composable security notion named Non-Interference (NI).

Definition 2.5 (d Non-Interferent (d-NI) [4])**.** A circuit G is d-NI if any set of at most $d' \leq d$ probes can be simulated with at most d' shares of each input.

Intuitively, the above model grants composable security since a probed value in a circuit can be simulated with an input share, which on its turn is the output share of a previous circuit. In case the latter circuit is also non-interferent, this output value can again be simulated with an input share. This chains until we reach the encoding function, the function sharing the plaintext or key. Since the adversary can only probe d values we need at least $d + 1$ shares to protect against our probing adversary. While the notion of non-interference is a good start and captures a composable security notion over the serial composition of circuits, the notion is not sufficient to provide protection when circuits are composed in parallel (e.g., when two circuits share the same input). To this end Barthe *et al.* introduced the notion of *Strong Non-Interference (SNI).*

Definition 2.6 (d-Strong Non-Interferent (d-SNI) [4])**.** A circuit G is d-SNI if any set of d_1 probes on its intermediate variables and every set of d_2 probes on its output shares such that $d_1 + d_2 \leq d$, the totality of the probes can be simulated by only d_1 shares of each input.

We note that intermediate variables can also be the input or output variables of the circuit.

When the above notion of non-interference is combined with a sharing scheme with a high enough passive threshold, the composable notion provides for probing security as defined by Ishai *et al.* [69].

Recently, an improvement to the non-interference model has been proposed by Cassiers *et al.* [27]. The authors propose a *probe-isolating non-interference* model which is weaker than the SNI requirement but still provides for arbitrary compositional security. The model can be used to improve the latency or randomness costs of maskings compared to using the traditional SNI notions.

2.6.3 The Bounded-Query Model

Further on in Chapter 5, we will consider a different security model from the simulation-based framework described above. As the non-interference framework requires information-theoretic security, it is often not possible to leverage certain cryptographic properties of maskings. To that end, we describe a new security model where the adversary has only a limited number of queries. We consider that a query loosely resembles a trace, meaning that a query spans one call to the masked cipher or primitive.

In the bounded-query model, the security of a circuit C with input k against a t-threshold-probing adversary will be quantified by means of a left-or-right security game as depicted in Figure 2.3. The challenger picks a random bit b and provides the oracle $\mathcal{O}^b$, to which adversary $\mathcal{A}$ is given query access. The adversary queries the oracle by choosing up to t wires to probe, we denote this set by $\mathcal{P}$, and sends it to the oracle along with the inputs k_0 and k_1. Note that we consider the input of the circuit to consist of both the plaintext and the key. The oracle responds by giving back the probed wire values of $C(k_b)$. After a total of q queries, the adversary responds to the challenger with a guess for b. For $b \in \{0, 1\}$, denote the result of the adversary after interacting with the oracle $\mathcal{O}^b$ using q queries by $\mathcal{A}^{\mathcal{O}^b}$. For left-or-right security, the advantage of the adversary $\mathcal{A}$ is then as defined as

$$\mathrm{Adv}_{t\text{-}\mathsf{thr}}(\mathcal{A}) = |\Pr[\mathcal{A}^{\mathcal{O}^0} = 1] - \Pr[\mathcal{A}^{\mathcal{O}^1} = 1]|\,.$$

We refer to this security notion as the *bounded-query probing model*.

If an arbitrary number of queries is allowed, the above security definition is equivalent to the simulation-based definition from Definition 2.4. Indeed, if the simulator simply evaluates the circuit for an arbitrary choice of the secret inputs, no adversary can distinguish the simulation from the real circuit with advantage higher than $\mathrm{Adv}_{t\text{-}\mathsf{thr}}(\mathcal{A})$. However, we note that the strong non-interference framework defined above is still useful since the above probing security does

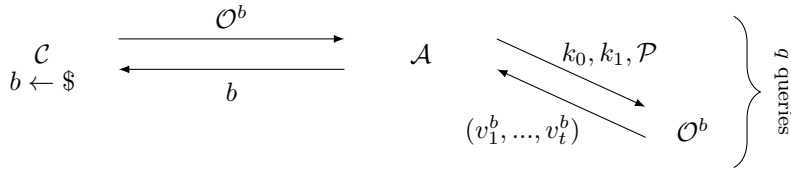


Figure 2.3: The privacy model for t-threshold-probing security consisting of a challenger $\mathcal{C}$, an adversary $\mathcal{A}$, a left-right oracle $\mathcal{O}^b$, two inputs k_0, k_1, a set of probes $\mathcal{P}$, and a set of probed wire values $(v_1^b, ..., v_t^b)$ of the circuit $C(k_b)$.

not necessarily allow for composition. Instead, we will use the left-right model to consider the security of the masked circuit in its entirety and thus security under composition need not be considered.

2.7 Examples of Masked Operations

We provide an example of an XOR and an AND operation and describe the interaction with the probing adversary and their security in the non-interference framework. The examples are based on a popular countermeasure called the Domain-Oriented Masking (DOM) scheme by Groß *et al.* [62]. A similar version was proven secure in the non-interference framework using glitch-extended probes by Faust *et al.* [59]. For simplicity, we only consider their first-order variants.

2.7.1 Masked XOR

We start with the XOR of shares. This operation is quite simple because it is linear in function of the Boolean masking operation. The operation is depicted in Figure 2.4. It operates by taking two shared inputs, for ease we consider only two shares, (a^1, a^2) and (b^1, b^2). The addition is done in one clock cycle where the shares are added share-wise giving the output

$$(c^1, c^2) = (a^1 \oplus b^1, a^2 \oplus b^2).$$

Probing Adversary We consider a first-order glitch-extended probing adversary. Since the entire masked XOR operation is calculated over one cycle, there are only two important probing positions. Namely, probing the outputs c^1 or c^2. The probes give back (a^1, b^1) and (a^2, b^2), respectively.

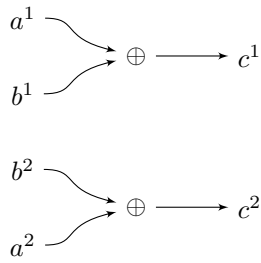


Figure 2.4: Depiction of a first-order masked XOR.

Non-Interference We show that the masked XOR is first-order non-interferent as from Definition 2.5. The operation is not strong non-interferent since probing the output gives back some input shares which were not re-masked with fresh randomness. As a result, a simulator which is given nothing can not perfectly simulate this output.

Showing that the operation is 1-NI is quite simple. We create a simulator which is given (a^1, b^1) when c^1 is probed and (a^2, b^2) when c^2 is probed. Given those shares the simulator can perfectly simulate the probed values.

2.7.2 Masked AND

The masked AND gate is more complicated compared to the XOR gate. The reason is because this operation is nonlinear in function of the Boolean masking operation. The operation is depicted in Figure 2.5. It operates on two inputs, each shared in two shares, (a^1, a^2) and (b^1, b^2). The operation works over two cycles. In the first cycle, all cross products between the shares are calculated $(a^1b^1, a^1b^2, a^2b^1, a^2b^2)$. Afterwards, a random bit r is added to the cross products a^1b^2 and a^2b^1 to create

$$(u^{1,1}, u^{1,2}, u^{2,1}, u^{2,2}) = (a^1b^1, a^1b^2 \oplus r, a^2b^1 \oplus r, a^2b^2).$$

The results are then stored in registers. In the second cycle, the cross products are added up to end up with two output shares

$$(c^1, c^2) = (u^{1,1} \oplus u^{1,2}, u^{2,1} \oplus u^{2,2}).$$

Probing Adversary In this example, a first-order glitch-extended probing adversary can place probes at several locations. We go over the different locations.

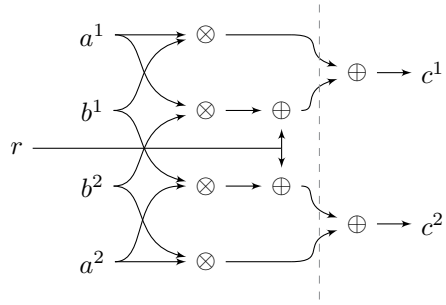


Figure 2.5: Depiction of a first-order masked AND. The dashed line represents a register stage.

- The adversary can probe one of the registers in the first cycle. Going from the top to bottom as depicted in Figure 2.5, the probe returns (a^1, b^1), (a^1, b^2, r), (a^2, b^1, r), or (a^2, b^2).
- The adversary can probe the output of the operation. The probe thus either returns $(u^{1,1}, u^{1,2}) = (a^1b^1, a^1b^2 \oplus r)$ or $(u^{2,2}, u^{2,1}) = (a^2b^2, a^2b^1 \oplus r)$.

Non-Interference The masked AND gate can be proven to be first-order strong non-interferent (1-SNI) in the glitch-extended probing model. For ease of the proof, we assume the output of the operation is stored in registers to stop glitches from propagating information outside the operation.

Take an arbitrary set of d_1 probes on the intermediate variables of the masked AND gate and d_2 probes on its output such that $d_1 + d_2 \leq 1$. We construct a simulator for the adversary's probed values which makes use of at most d_1 shares of the secrets a, b. We classify the internal variables in the following groups where we bundle all duplicated share.

1. a^i, b^i
2. $u^{i,j}$
3. c^i

We define two sets of indices I and J such that $|I| \leq d_1$, $|J| \leq d_1$ and the probed values can be perfectly simulated given only the knowledge of $\{a^i\}_{i \in I}$ and $\{b^j\}_{j \in J}$. The sets are constructed as follows.

- For every probe in groups $(1), (3)$ add i to I and to J.

- For every probe in group (2) add i to I and j to J.

Since the adversary is allowed to make at most d_1 internal probes, we have $|I| \leq d_1$ and $|J| \leq d_1$. We now show how the simulator behaves for the internal observed values.

- For each observation as in group (1), by definition of I and J the simulator has access to a^i and b^i. Thus we can perfectly simulate the values.
- For each observation as in group (2), the simulator has access to a^i and b^j, and r can be simulated as a uniform random variable if it was not simulated before. Thus, we can perfectly simulate the $u^{i,j}$ value.
- For each observation as in group (3), by definition $i \in I, J$. We assign a random value to the term $u^{i,j}$, with $j \neq i$. Since r is not viewed by any other probe and this is perfectly simulated as a uniform random value.

We now simulate the outputs c^i using no information from the inputs.

Following the operation of the masked AND gate, each output bit consist of the random bit r. Since the adversary only has one probe, this random value was not probed before. Thus, the simulator can assign a random and independent element to c^i, completing the proof.

2.8 Conclusion

In this chapter, we have studied masking as a countermeasure against side-channel analysis. It is possible to create maskings at gate level, meaning that we can create a secure XOR gate and a secure AND gate. This allows us to secure any operation making it possible to create compilers which automatically secure your implementation. Nevertheless, masking comes at a cost. They can significantly increase the area and latency of applications. We have also seen that each AND gate needs some fresh randomness to secure it. In practical application this means that thousand to tens-of-thousands of bits of randomness are used to compute the masking of a symmetric primitive. The generation and cost of this randomness is often not described in the literature.

Chapter 3

Generating Randomness in Maskings

In this chapter, we introduce the role of randomness in maskings, we study how to generate it, and the cost related to this generation. We also show that non-cryptographic methods for the generation of randomness might lead to insecurities.

This chapter is written by myself but is based on previous works, apart from Section 3.5 which contains unpublished novel ideas.

3.1 The Cost of Composable Security

In the previous chapter, we have seen maskings as a countermeasure against side-channel analysis. Maskings can also secure arbitrary computations as they can be made on a gate level. However, from our examples we have also seen that each secure AND gate requires fresh randomness. In fact, in this section we show that at some point a high degree operation *must* use randomness in order to warrant arbitrary composable security.

We first start with the assumption that the security game we are playing adheres to Definition 2.4 on simulation. More general, consider a simulation security game as in Definition 2.4 with a black box simulator, meaning the simulator can be given arbitrary information (except for secrets). We focus on the distinguisher which is given all the inputs of the circuit but not the randomness generated

in the circuit. This distinguisher needs to be given the input of the circuit in order to achieve arbitrary composable security. For the sake of the following theorem, we call this security model a "composable simulation model".

We formally show that a high degree circuit always requires fresh randomness in order to be secure in the above composable simulation model. The following proof is based on the result from Petrides [85] on how many shares are needed to achieve non-completeness.

Theorem 3.1. *Given a circuit G which computes a degree t operation working on s input and output shares. In case $t > (s-1)/d$, G always requires fresh randomness to secure against a dth-order probing adversary in a "composable simulation model".*

Proof. We show the above by assuming G does not use fresh randomness. From Petrides [85, Proposition 2.5], we find that, since $t > (s-1)/d$, there is no d^{th}-order non-complete sharing of a degree t function with s shares. As a result, there are d outputs of G whose computation depend on all shares of an input secret. Taking these outputs as probe positions, the distinguisher in the security game can perfectly recreate these d wires. However, the simulator can only simulate these values when it is given the input secret which results in a contradiction to the assumption of the security model. As a result, fresh randomness must be used in order to secure G in the composable security model. □

In particular, the above theorem holds for any d^{th}-order (S)NI or PINI circuit. As a result, these frameworks will always require fresh randomness since calculating a symmetric primitive inadvertently means calculating a high degree function. The generation or cost of this randomness as well as the integration of the random number generator is, however, almost never discussed in the literature. Instead, the amount of fresh randomness is seen as an abstract cost. In the rest of this section, we try giving a picture how one can securely integrate a random number generator along with the cost and requirements of this generation.

3.2 Probing Robust Random Generators

We first tackle the question on the integration of random number generators in the masking. The plug-and-play use of a random number generator does not guarantee security. Consider using a cryptographic primitive (a stream cipher) to generate randomness for the masking, this scenario can still allow for attacks when the adversary can probe the stream cipher as well as the masked state

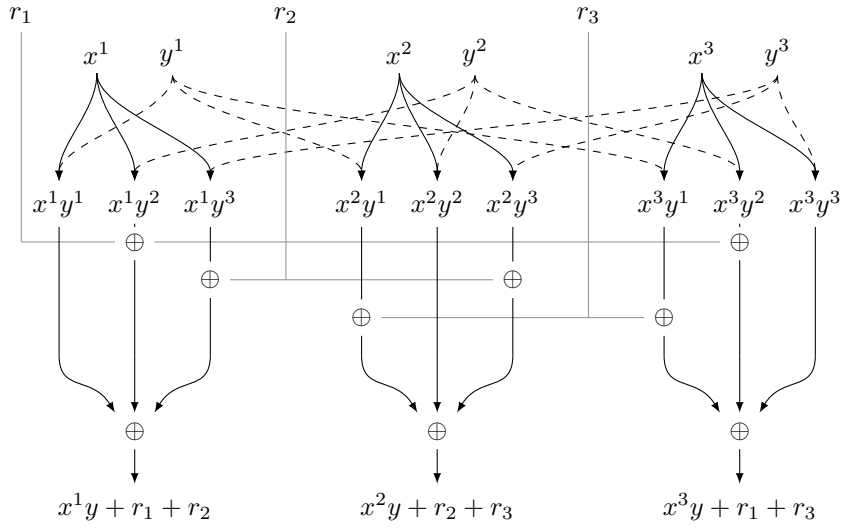


Figure 3.1: A second-order secure AND gate.

function in a higher-order attack. We give a simple example. Recall the secure AND gate seen in Section 2.7.2. We consider a second-order secure variant. In this variant, three cross products are summed using two random bits. This is depicted in Figure 3.1.

A probing adversary discovers both r_1 and r_2 with one probe in the stream cipher (for example if r_1 and r_2 are the output of the same S-box). The second probe is then placed in the first XOR at the end in Figure 3.1 revealing the input secret y thus breaking the second-order secure masking.

We observe that embedding a random number generator in a masking is not trivially secure. Luckily, there are some positive answers. We quickly take a look at "probing robust random generators" introduced by Ishai *et al.* [67] in 2013. Considering probing security from Section 2.6.1, we require that in case our masking is d^{th}-order probing secure, integrating the random number generator does not change that property. Achieving this is not difficult, even for arbitrary maskings, although it can be costly. We introduce a specific method to create a robust random generator by using multiple random generators. Since this thesis focuses on first and second-order security, using multiple generators is quite efficient.

First, we consider using ideal random number generators, deterministic or non-deterministic, where each output is independently uniformly distributed. When

plugging an ideal generator to a first-order masking the entire construction is still first-order probing secure. Namely, either the adversary probes the masking, which was presumed to be first-order secure, or the random generator whose computation does not depend on the secrets in the masking. This makes integrating random generators in masking, from the view of first-order probing security, quite trivial. For second-order security, a similar thing is done by taking two ideal generators where we sum their output together for use in the masking. Again, a second-order probing adversary gains nothing by placing the two probes in either the masking or the generators. But notice that placing one probe in the generator only recovers at most one random bit which is used in the masking thanks to the second (unprobed) generator. This one random bit could have been simulated by placing a probe in the masking itself, thus the security is the same as placing two probes in the masking. In other words, no matter where the adversary placed the probes, the construction remains second-order probing secure.

The above construction only holds for probing security. One could still break the construction in practice. For example, an adversary could attack a first-order scheme by attacking the non-masked pseudo-random number generator before attacking the masking. For example, De Meyer [44] showed how one can break an AES in counter mode in only 256 traces without knowing its output. That means that if one used the AES to generate the randomness for the masking and this AES was called 256 times, the seed of the generator might be recovered. In practice, that means that the designer typically needs to add some non-deterministic randomness (which has no recoverable seed) to these pseudo-random number generators to counter such attacks. In the theoretical world, it is still an open question how to make robust random generators in other, more realistic, security models.

3.3 Outline of Non-Deterministic Random Number Generation

In the previous section, we assumed to have access to ideal random number generators. However, this does not exist in practice. Instead, we take a look at how to generate randomness in reality.

In cryptography we distinguish two kinds of randomness, non-deterministic randomness and pseudo-randomness (deterministic randomness). We start with the generation of non-deterministic randomness which is done using physical sources. We base the explanation of non-deterministic random number generators using the SP 800-90 [81] publication by the National Institute of

Standards and Technology (NIST). Essentially, a non-deterministic random number generator consists of two parts, an entropy source and a processing function which is typically a cryptographic function.

The entropy source, in this case noise source, is a way to turn a physical property into a digital value. These physical properties could for example be heat variations of the environment, the activity of your mouse on a laptop, or entropy created by using a quantum physical effect. The process of generating randomness from a physical source and the question whether it is indeed "random" is generally complex and is outside the scope of this thesis.

A popular method for generating randomness on hardware is via the use of race-conditions. Environmental factors influence the switching behaviour of logic gates. This switching behaviour can be made probabilistic and dependent on its environment such that it becomes possible to generate random bits. As these bits originate from physical effects instead of the output of some mathematical function, we call this *non-deterministic randomness*.

Nevertheless, an adversary could have control over the environment from which the randomness is extracted. Think of heating up or cooling down the targeted device. To prevent such attacks, *health tests* are added to the entropy source. These verify certain simple statistical properties such as the mean or variance of the output of the noise source.

The second component of a random number generator is the processing function. This takes the outputs of the entropy source, which are typically not of full entropy, and processes them. This processing function can be made XORing some entropy bits or by using a cryptographic primitive such as a hash function. Again health tests are added to the output of the processing function to verify whether the assumption of a full entropy output is met.

It is difficult to provide the cost of non-deterministic random number generation for use of arbitrary maskings as its cost is dependent on the used technology. However, non-deterministic random number generation is generally extremely expensive. Maskings often operate at high frequency due to their need of registers to protect against glitches. As a result, a heavy-duty random number generator might not even make five let alone ten random bits per cycle of a masking. Regular maskings of the AES often require twenty, for first-order security, or fifty, for second-order, random bits per cycle. Instead, non-deterministic random number generators are often used to load the seed of a deterministic random number generator which can provide pseudo-random bits at a much more efficient rate.

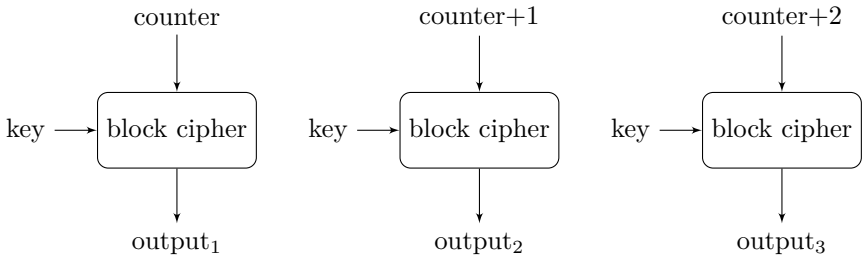


Figure 3.2: Counter mode of operation to transform a block cipher into a stream cipher providing birthday bound security.

3.4 Generation Using Cryptographic Primitives

In this section, we investigate how to stretch a seed of non-deterministic randomness for the fresh randomness needed in a masking. This generation of pseudo-randomness has been investigated thoroughly by the cryptographic community. For example, one can use hash functions or block ciphers to create stream ciphers. An example is the counter mode of operation as depicted in Figure 3.2.

The use of a cryptographic primitive results in a practical secure masking. Meaning that if one considers a perfect secure d^{th}-order secure masking, then one can replace the ideal randomness source by the cryptographic primitive and still be guaranteed security against a computationally bounded adversary. However, such generators come at a significant cost. We have bundled some of the costs in Table 3.1. We do note that these costs are bundled from various sources, thus the performance comparison is not optimal as the implementations were not generated using the same libraries. However, they still give an intuition on the cost.

Table 3.1: The cost of generating randomness using cryptographic primitives in Gate Equivalent (GE). The security parameter provides the claimed security of the underlying primitive.

Primitive	Security	Area Cost	Bits per Cycle	Area Cost per Bit
AES [82]	128-bit	118440GE	128	925GE
Grain-128a [1]	128-bit	5168GE	32	161GE
PRINCE [19]	128-bit	8263GE	64	129GE
Trivium [23]	80-bit	5504GE	64	86GE

We now take two of the most popular countermeasures to protect the AES against first and second-order attacks. These are the countermeasures from Groß *et al.* [63] and De Meyer *et al.* [47]. We compare their area cost using [47, Table 5] (we neglect the latency as it is almost equivalent for all schemes). We then apply the cheapest solution, in this case Trivium, to generate the needed randomness. Using Table 3.1, we estimate the additional area cost. The results are shown in Table 3.2.

Table 3.2: We take the schemes by Groß *et al.* and by De Meyer *et al.* and provide their original area cost (using the Nangate 45nm library) along with the additional area cost to generate the randomness using Trivium from Table 3.1

Scheme	Security Order	Original Cost	Additional Cost
Groß *et al.* [63]	1	7337GE	1548GE
De Meyer *et al.* [47]	1	6557GE	1634GE
Groß *et al.* [63]	2	12024GE	4644GE
De Meyer *et al.* [47]	2	10931GE	4558GE

The above costs for randomness generation are optimistic. For example, the first order sharing by De Meyer *et al.* reports using the 19bit/cycle variant of Trivium which requires over 4000GE. Moreover, recall that we need to double the cost for second-order secure schemes in case we want to have a probing robust random generator. Even with these optimistic numbers, we observe that around 15-30% of the total area cost is dedicated to the generation of randomness. It is important to note that this cost also scales with the architecture, meaning the cost increases proportionally when round-based architectures would be used. In other words, a round based version of the sharing by De Meyer *et al.* would require 26144GE for first-order and 72928GE for second-order security to generate the randomness. The randomness also quadratically scales with the protection order, as can be seen in the work by Groß *et al.*. The authors show their masked S-box requires $9d(d+1)$ random bits, with d the protection order. From the cost of Trivium, we can thus estimate the area cost to generate randomness to be $774d(d+1)$GE.

3.5 Generation Using LFSRs: Three Attacks

In practice, random number generators using cryptographic primitives are not used due to their high cost. Instead, non-cryptographic generators are implemented. An example is the Linear Feedback Shift Register (LFSR), a depiction is given in Figure 3.3. This is a linear system which updates itself

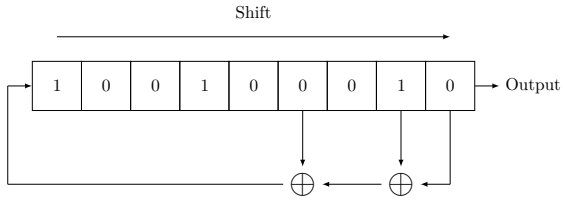


Figure 3.3: Example of a linear shift feedback register. Here the taps are placed on the ninth, eight, and sixth registers.

by shifting the entire state and XORing the output with other state elements (these are called the *taps*) to create the new state element. The *cycle length* of an LFSR is determined by how many applications it takes to output the same element again.

These LFSRs have a significantly reduced cost. We quickly sum this up in Table 3.3. Essentially, the cost to generate a bit of randomness is one register and an XOR.

Table 3.3: The cost of some two-tap LFSRs. We estimate the cost of a register by 5GE. The number of bits the LFSR can generate per cycle is calculated from the number of state registers which are unaffected by the taps.

Size	Area Cost	Bits per Cycle	Area Cost per Bit
31-bit	156GE	1	156GE
31-bit	201GE	29	6GE
63-bit	409GE	61	6GE
127-bit	825GE	125	6GE

In this section, we show that the use of LFSR's to generate the randomness in maskings can lead to attacks. We provide three attacks against the sharings from Groß *et al.* [63] and De Meyer *et al.* [47]. We limit our analysis to theoretic arguments and attacks in the probing model.

3.5.1 Attack 1: Direct Probing

The first attack is the most simple one. We take the second-order version of the masked AES from Groß *et al.* and assume the randomness is generated by a n-bit LFSR with maximum length. We first start with the following easy lemma.

Lemma 3.2. *An LFSR with maximum length and initial state* $(r_1, ..., r_n)$ *generates the bit* $r_i + r_j$ *for any* $i, j \in \{1, ..., n\}$ *and* $i \neq j$.

Proof. Since the LFSR has a maximal length, it cycles through all $2^n - 1$ states within the shift register. Since only $2^n - 1$ different combinations between $(r_1, ..., r_n)$ are possible (excluding the all zero case), each combination of the form $r_i + r_j$ for any $i, j \in \{1, ..., n\}$ and $i \neq j$ will be generated. □

Consider again the AND gate from Figure 3.1. The attack is similar to the one described in Section 3.4. One can place a probe in the first XOR at the end to reveal the value $x^1y + r^1 + r^2$. However, from the above lemma, we also know that the LFSR will eventually output the bit $r_1 + r_2$. Thus, one can place a probe in the masked cipher where this bit is used to reveal the secret y.

When using a two-tap LFSR with the taps next to each other (these LFSRs are popular in practice), the values $r_i + r_{i+1}$ are generated as one of the first outputs making it possible to only attack the first round of the masked primitive thus avoiding the need for template attacks and instead being able to directly use Differential Power Analysis (DPA).

The attack not only works for the masking of Groß *et al.*, but for most of the SNI-secure operations following the original design of Ishai, Sahai, and Wagner [69].

3.5.2 Attack 2: Misalignment

The second attack is focused on the masking by De Meyer *et al.*. Before we start, we note that the original masking uses the stream cipher Trivium to generate the fresh randomness. Instead, we replace the cryptographic primitive by an LFSR and verify the security. In other words, this attack does not invalidate the masking itself, it only excludes the possibility of using an LFSR to generate its randomness.

We consider the first-order variant of the masked S-box, this is depicted in Figure 3.4. We first note that the input secret of the multiplier x can not be equal to zero, neither can the random value r_2. The details on how this is assured can be found in the original article [47].

Consider the second share of the output of the masked S-box $x^{-1} + r_2r_3$. Then, consider that the random bytes r_1 and r_2 are generated by the same LFSR L. Meaning that there are indices i and j such that

$$r_2 = (r_{2,1}, ..., r_{2,8}) = (L^i(s), ..., L^{i+7}(s))$$

and

$$r_3 = (r_{3,1}, ..., r_{3,8}) = (L^j(s), ..., L^{j+7}(s)),$$

with s the initial state of the LFSR. Since the randomness r_2 and r_3 are generated in different cycles, we can consider $j > i + 8$. As a result, it is highly likely that the system $(L^i(s), ..., L^{i+7}(s), L^j(s), ..., L^{j+7}(s))$ is not linearly independent. In other words, (r_2, r_3) will not be joint uniform as some bits will be linear combinations of others. Thus, the multiplication $r_2 r_3$ will not be uniform and the mutual information of $x^{-1} + r_2 r_3$ and x^{-1} for random bytes r_2, r_3 is non-zero. We have tested this non-uniformity using software and various different LFSRs.

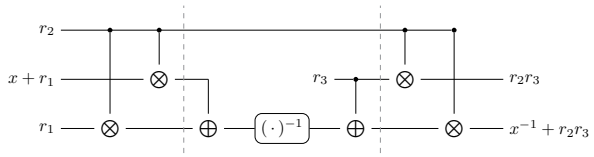


Figure 3.4: First-order sharing of De Meyer *et al.* [47].

The same attack can be performed on the second-order version of the scheme as shown in Figure 3.5. In this case, we expect $r_3 r_4 r_5 + r_3 r_6$ to be non-uniform. Again, this would be a first-order attack showing that using an LFSR does not allow the masking to provide a significant increase in in function of its order.

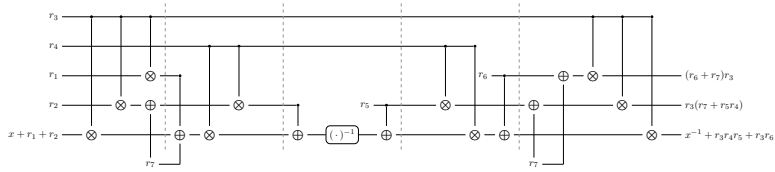


Figure 3.5: Second-order sharing of De Meyer *et al.* [47].

3.5.3 Attack 3: State Retrieval

The third attack again focuses on the masking by De Meyer *et al.*. However, this attack works on any random number generator with a small state size. We sketch the outline of the attack.

Consider the multipliers providing the outputs $r_2 r_3$ and $r_1 r_2$ in the first-order masking as shown in Figure 3.4. This output only depends on the output of the random number generator in the masking. Considering there are 200 S-boxes in

the AES-128 cipher, each trace thus contains 400 data points providing leakage on the state of the random number generator. These data points could be used to guess the state using a horizontal-like attack, see the work by Clavier *et al.* [32]. The computational cost of the attack thus depends on the size of the state.

Performing a state guessing attack using only 400 data points is difficult. However, we can equivalently attack the second-order scheme as it contains 1200 exploitable points. Once the state is retrieved, the attacker can strip away the randomness to have a trace of an unmasked AES. An improved version of the attack combines these two steps in one to reduce the number of traces needed for the attack to succeed.

The attack is interesting as it has a high computational complexity but low data complexity. It shows that there might be cryptanalytic attacks on maskings requiring only a few traces by abusing the random number generator in the masking. This attack would also not be noticed by t-tests as we combine hundreds of data points within the same trace.

3.6 Conclusion

In this chapter, we have discussed the need and role of randomness in maskings and how to generate it. We have seen that simulation-based models such as non-interference always require fresh randomness to secure circuits. We have discussed randomness generation using non-deterministic generators and how they provide the seed for deterministic generators. These deterministic generators can be made using cryptographic primitives in which case they come at a high cost. However, using non-cryptographic generators comes at a potential security risk.

One can think that our attacks on LFSR's are not realistic because LFSR's are probably too weak to be used in practice. However, these attacks pertain to the current academic literature. We find that in several works [72, 94, 100], a 31-bit LFSR is used to generate the randomness of the maskings. The above attacks show the danger of using an LFSR to generate this randomness. However, discussion with the authors showed that instead they use a separate LFSR for each bit generated per clock cycle. Nevertheless, as shown in Tables 3.1 and 3.3 the cost of an LFSR to generate only 1 bit per cycle can be higher than the cost of using a cryptographic primitive. We thus argue that the use of LFSRs either leads to an insecure or an inefficient masking.

We are faced with a problem when dealing with randomness in maskings as

we find that this always comes at a high, and often neglected, cost. This creates an interesting open problem, can we create maskings without using fresh randomness by looking at different security models as opposed to non-interference? In the next chapter, we give a positive result where we start investigating how we can significantly reduce the randomness cost without affecting the cost of the masking.

Chapter 4

Threshold Implementations

In this chapter, we introduce a special class of masked functions called threshold implementations. These maskings were introduced by Nikova *et al.* [84] in 2006. We discuss why this class of masked functions is interesting for the thesis, provide their definition, prove their security in the probing model, and provide a threshold implementation of the AES.

Theorem 4.3 and Section 4.4 are based on a work published at TIS at CCS [52] which was supervised by Svetla Nikova and Vincent Rijmen. Sections 4.5 and 4.6 are written by myself and contain unpublished novel research. The rest of the chapter is based on previous work.

4.1 Motivation

As we have seen in the previous chapters, masking can be an effective countermeasure against side-channel attacks. However, this comes at a price. Masking schemes not only increase costs in terms of area, latency, or energy consumption but they require a significant number of random bits to be injected into its calculation. The security requirements of this generation are currently unknown and its, cryptographic, strong generation is expensive. In order to improve the trust in masking schemes and reduce their cost, we turn towards methods which allow the reduction of the needed randomness in the scheme.

In 2006, Nikova *et al.* [84] defined three properties for masked functions; correctness, non-completeness, and uniformity. In short, these properties ensure the masking is sound, viable for use in hardware, and releases the need of fresh

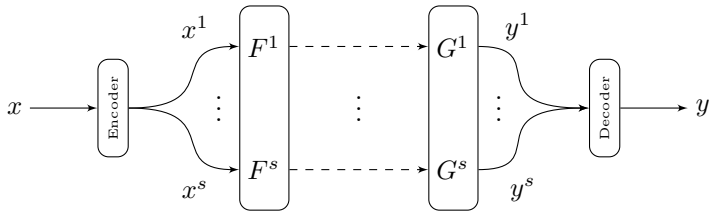


Figure 4.1: Representation of a threshold circuit.

randomness, respectively. It is this last property which is of interest for the thesis. The use of uniform masked functions allows the creation of masked primitives which only require randomness to share its input. This leads to a significant reduction of randomness. To give an example, a PRESENT-128 cipher masked with a non-uniform sharing would require 2300 random bits per block [26], while a threshold implementation of PRESENT only requires 192 random bits [93], both numbers include the randomness needed to share the plaintext and key. Even in this simple example, threshold implementations give an over tenfold improvement in randomness cost.

4.2 Definitions

A threshold implementation consists of several layers of Boolean functions, as shown in Figure 4.1. As for any shared design, a black-box encoder function generates a uniform random sharing of the input before it enters the shared circuit and the output shares are recombined by a decoder function. At the end of each layer, synchronisation is ensured by means of registers which are assumed to stop glitches.

We go over the three properties of threshold implementations as introduced by Nikova *et al.* [84]. The first property pertains to the secret sharing of a variable. We require that a sharing provides the expected threshold level, meaning that with $d+1$ Boolean shares the adversary does not get information on the sharing's secret with less than $d+1$ shares. This leads to the property of a uniform sharing.

Definition 4.1 (Uniform Masking [84])**.** A masking $\mathbf{X}$ in s_x shares is uniform if for all $x \in \mathbb{F}$ we have

$$\Pr(\bar{X} = \bar{x} \,|\, X = x) = \begin{cases} |\mathbb{F}|^{-s_x+1} & \text{if } \bar{x} \in Sh(x)\,, \\ 0 & \text{else}\,. \end{cases}$$

In words, each share vector has an equal chance to occur.

We aim to create an algorithm which works over shared variables instead of its secrets. As such, we denote $\bar{F}(\bar{x}) = \bar{y}$ as the shared function of $F(x) = y$. This shared function takes in the shares of x and gives back a sharing of F's outputs. We can thus consider a sharing of F as functions F^i taking in shares of x and giving a share of y as output. We first require that the shared function $\bar{F}$ gives a correct sharing of the output y, meaning that the sum of the output shares $\sum_{i=1}^{s_y} y^i$ is equal to y. This leads to the correctness property.

Definition 4.2 (Correctness)**.** The sharing $\bar{F}(\bar{x})$ is a correct sharing of F when $\sum_i F^i(\bar{x}) = F(\sum_i x^i)$.

Due to the effect of glitches, a sample of a power trace can contain more information than of just one share and instead contains joint information on all gates viewing a glitch. To this effect harming the privacy of a secret shared algorithm, we need to carefully place registers and demand that each computation is made using *non-complete* information of a secret. This leads to the definition of non-completeness on the level of coordinate functions.

Definition 4.3 (Non-completeness [84])**.** A shared function $\bar{F}(\bar{x})$ is non-complete if each of its shares F^i uses at most $s_x - 1$ input shares.

The above definition secures maskings against the effect of glitches provided the input and of each non-complete function is synchronised.

Lastly, we require that all sharings in an algorithm contain enough entropy, meaning that each sharing is uniformly distributed as in Definition 4.1. Since we assume that the initial sharing of secrets introduces enough entropy such that the sharing is uniform, we require that all subsequent functions computing on a uniform sharing give back an output which is again uniform. This property is given in the following combinatorial form.

Definition 4.4 (Uniformity [84])**.** A shared function $\bar{F}(\bar{x}) = \bar{y}$ is uniform if $\forall x \in \mathbb{F}$, $\forall \bar{y} \in Sh(F(x))$:

$$\left| \left\{ \bar{x} \in Sh(x) \,\middle|\, \bar{F}(\bar{x}) = \bar{y} \right\} \right| = \frac{|\mathbb{F}|^{s_x - 1}}{|\mathbb{F}|^{s_y - 1}} .$$

We note that if the number of inputs and outputs of the shared function are equal, the above property is equivalent to the shared function being a permutation.

A uniform function maps a uniform input to a uniform outputs, this is proven by Bilgin *et al.* [16].

Lemma 4.1 (Theorem 3 [16])**.** *If a shared function $\bar{F}$ is uniform then its output is uniform provided its input is uniform.*

4.3 Security in the Probing Model

In the original work by Nikova *et al.*, there is a proof showing that the mean power consumption of a threshold implementation can not be related to any secret variable processed. While this proof gives a tangible connection to the real-world security of threshold implementations, the current standard security model is the probing model. As such, we rewrite the original proof and show that a threshold implementation is first-order glitch-extended probing secure.

We recall the probing security model from Section 2.6.1. In this adversary model, a probe can essentially view the inputs from one share of a function in the threshold implementation. In the security model, we have to show that a simulator can simulate these probed values from scratch. This comes down to showing the probed values are indistinguishable from uniform random.

To give such a proof of security, we first require the following lemma proven by Bilgin *et al.* [16].

Lemma 4.2 (Lemma 1 [16])**.** *Given a uniform masking $\bar{X}$, then for all $i \in \{1, ..., s_x\}$,*

$$\Pr(\bar{X}^{\bar{i}} = \bar{x}^{\bar{i}}) = |\mathbb{F}|^{1-s_x}.$$

With $\bar{x}^{\bar{i}}$ denoting the vector of all shares except x^i.

In words, all shares but one of a uniform masking are uniform random.

We now show a threshold implementation is first-order glitch-extended probing secure. We slightly abuse notation by denoting a threshold implementation as a mathematical construction consisting of the composition of non-complete and uniform shared functions.

Theorem 4.3. *A threshold implementation consisting of correct, non-complete, and uniform shared functions is first-order glitch-extended probing secure.*

Proof. First, due to each shared function being correct, the entire design is also correct. As a result, a threshold implementation is sound.

Consider an arbitrary share of a function is probed. Denote the shared function by $\bar{F}(\bar{x})$ and the probed share by F^i. Due to the input encoder of a threshold implementation providing a uniform input sharing and due to each function in the threshold implementation being uniform, we can conclude from Lemma 4.1 that $\bar{x}$ is uniform. Since $\bar{F}$ is non-complete, we know there exists a $j \in \{1, ..., s_x\}$ such that the glitch-extended probe returns the values $\bar{x}^{\bar{j}}$. From Lemma 4.2, we know that

$$\Pr(\bar{X}^{\bar{j}} = \bar{x}^{\bar{j}}) = |\mathbb{F}|^{1-s_x}.$$

As a result, we can simulate the probed values by uniform randomness. Thus, the threshold implementation is also first-order private concluding the proof. □

4.4 Reducing Randomness

We have learned what threshold implementations are and we have shown that they are secure, we now move on to show why they are useful in light of the thesis. First, note that the definitions nor the proof of security mention fresh randomness. Contrast this with the definition of SNI from Section 2.6.2, where randomness is defined as a resource which is not given to the distinguisher. This already provides the intuition that threshold implementations give more relaxed properties compared to the SNI properties. This can in fact be made formal. We prove that if a shared function which can be proven to be non-interferent is given a uniform input, it will give back a uniform output.

We recall the strong non-interference model from Section 2.6.2 and use the probabilistic definition given in the work of De Meyer *et al.* [46]. We then generalise first-order non-interference to work over an arbitrary number of Boolean masked shares (previously only 2 shares were considered). We require a simulator who can reproduce each possible probe or sets of $s_y - 1$ output shares given a set of $s_x - 1$ input shares. Intuitively, this generalisation of non-interference grants composable security since a probed value in a function can be simulated with all-but-one input shares, which in its turn forms the output of a previous function. In case the latter function is also non-interferent, these output values can again be simulated with all-but-one of its input shares. This chains until we reach the initial sharing function. Due to the passive threshold of the Boolean masking scheme, knowing all-but-one shares of the encoder's output implies that the adversary does not learn the secret of the shares.

We thus look at our generalised notion of first-order non-interference in case the considered function is given uniform inputs.

Definition 4.5 (First-order Non-interference)**.** A shared function $\bar{F}(\bar{x}) = \bar{y}$ with uniform input shares is 1-NI if, for any glitch-extended probe q and any $\bar{x}$, the following condition holds:

$$\exists i \in \{1, ..., s_x\} : \Pr(Q = q \,|\, \bar{X} = \bar{x}) = \Pr(Q = q \,|\, \bar{X}^{\bar{i}} = \bar{x}^{\bar{i}}) \,,$$

together with:

$$\forall j \in \{1, ..., s_y\},\ \exists i \in \{1, ..., s_x\} : \Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}} \,|\, \bar{X} = \bar{x}) = \Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}} \,|\, \bar{X}^{\bar{i}} = \bar{x}^{\bar{i}}) \,,$$

with $\bar{*}^{\bar{i}}$ denoting all shares except for the ith-share.

Using the above property, we show that a non-interferent share function is also uniform.

Theorem 4.4. *If a shared function $\bar{F}(\bar{x}) = \bar{y}$ is 1-NI it is uniform.*

Proof. We take an arbitrary secret x with $F(x) = y$ and an arbitrary $\bar{y} \in Sh(y)$, and we assume that $\bar{X}$ is uniform. For a uniform input sharing $\bar{X}$, it follows from Definition 4.5 that every set of all-but-one output shares $\bar{Y}^{\bar{j}}$ is independent of the input secret X since for an arbitrary x we have

$$\begin{aligned}
\Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}},\, X = x) &= \sum_{\bar{x} \in Sh(x)} \Pr(\bar{X} = \bar{x})\, \Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}} \mid \bar{X} = \bar{x}) \\
&= \sum_{\bar{x} \in Sh(x)} \Pr(\bar{X} = \bar{x})\, \Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}} \mid \bar{X}^{\bar{i}} = \bar{x}^{\bar{i}}) \\
&= \sum_{\bar{x} \in Sh(x)} \Pr(\bar{X}^{i} = \bar{x}^{i})\, \Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}},\, \bar{X}^{\bar{i}} = \bar{x}^{\bar{i}}) \\
&= \sum_{\bar{x}^{\bar{i}}} \Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}},\, \bar{X}^{\bar{i}} = \bar{x}^{\bar{i}})\,,
\end{aligned}$$

which is independent of the secret X.

We prove that the output of $\bar{F}$ is uniform. We find the following equalities for an arbitrary j.

$$\begin{aligned}
\Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}},\, Y = y) &= \sum_{F(x)=y} \Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}},\, X = x) \\
&= \sum_{F(x)=y} \Pr(X = x)\, \Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}} \mid X = x) \\
&= \sum_{F(x)=y} \Pr(X = x)\, \Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}}) \\
&= \Pr(Y = y)\, \Pr(\bar{Y}^{\bar{j}} = \bar{y}^{\bar{j}})
\end{aligned}$$

□

Oppositely, there are shared functions which adhere to the threshold implementation properties but which are not 1-NI. In fact, it seems the sharings which require no fresh randomness belong to this class. A simple

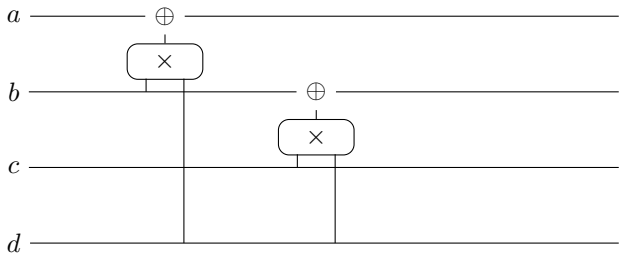


Figure 4.2: This figure depicts the Feistel structure in the quadratic class $\mathcal{Q}^4_{294}$.

example is given from the quadratic class $\mathcal{Q}^4_{294}$ given as a lookup table as $F(a,b,c,d) : 0123456789BAEFDC$ and in ANF as

$$x = f(a,b,c,d) = bd + a\,, \qquad y = g(a,b,c,d) = cd + b\,,$$
$$z = h(a,b,c,d) = c\,, \qquad t = k(a,b,c,d) = d\,.$$

A non-complete and uniform sharing is given by

$$x^1 = b^1d^1 + b^1d^2 + b^2d^1 + a^1 \quad y^1 = c^1d^1 + c^1d^2 + c^2d^1 + b^1 \quad z^1 = c^1 \quad t^1 = d^1$$
$$x^2 = b^2d^2 + b^2d^3 + b^3d^2 + a^2 \quad y^2 = c^2d^2 + c^2d^3 + c^3d^2 + b^2 \quad z^2 = c^2 \quad t^2 = d^2$$
$$x^3 = b^3d^3 + b^3d^1 + b^1d^3 + a^3 \quad y^3 = c^3d^3 + c^3d^1 + c^1d^3 + b^3 \quad z^3 = c^3 \quad t^3 = d^3$$

It is clear the above sharing is non-complete. Observe that the quadratic class admits a Feistel structure, this is depicted in Figure 4.2. Namely, the output is constructed by taking b and d multiplying them and adding them to a, and multiplying c and d and adding them to b. Similarly, the sharing admits a similar Feistel structure since all shares of c and d are mapped directly to the output. As a result, the above sharing is a permutation and thus uniform.

However, the above sharing is not 1-NI following Definition 4.5 as

$$\Pr(X^{\bar{3}} \mid \bar{A}, \bar{B}, \bar{D}) \neq \Pr(X^{\bar{3}} \mid \bar{A}^{\bar{3}}, \bar{B}^{\bar{3}}, \bar{D}^{\bar{3}})\,,$$

since x^2 also depends on b^3 and d^3.

Similar constructions for other quadratic classes can be found in the work by Bilgin *et al.* [17]. Recently, by achieving uniformity over two register stages, new sharings using fewer shares were found by Shahmirzadi and Moradi [93].

4.5 The Necessity of Uniformity

In the work by De Meyer *et al.* [46], an example is given to show that the property of uniformity is not necessary for first-order probing security. We believe this example has led to confusion in the community resulting in people being eager to dismiss the notion. To clarify, the example given by De Meyer *et al.* is correct but their conclusion has been posed too vague.

To start, that uniformity is not necessary has already been argued in the original work by Nikova *et al.* [84]. They show that a non-complete and correct masking is already secure against a side-channel adversary. Uniformity is useful when composing non-complete sharings and be assured of the protection of the whole.

The example given by De Meyer *et al.* shows two specific non-uniform circuits for which the composition between each other is still first-order secure. Thus, the conclusion of uniformity not being necessary is drawn. However, examples for this conclusion were already common knowledge. Take the AND gate from Section 2.7.2, variations of which go back to 2003 by Ishai *et al.* [69], the first stage of this design is correct and non-complete but not uniform. Nevertheless, in the same section we showed that the two-stage design is probing secure. The conclusion of the counterexample was that uniformity was not necessary for composition. We conclude this more formally.

There exist two probing-secure masked circuits C_1, C_2 which are not uniform for which the composition $C_2 \circ C_1$ is again probing secure.

The above result should not be exaggerated. We emphasise that we have no knowledge of a secure masking which at no point achieves uniformity. The only example known in the literature is the work by Daemen [34] proposing a non-uniform sharing of an S-box which can be used in a cipher. However, this solution has not been proven to be secure. We again repeat the previous result, Theorem 4.4, that any non-interferent secure solution is also uniform. Meaning, the solutions searched for in the work by De Meyer *et al.*, namely NI-secure ones, all achieve uniformity.

We try to rephrase the result by De Meyer *et al.*, uniformity is not a necessary property when verification tools can be used to ensure the probing security of a circuit as a whole. The relaxation of uniformity can lead to efficiency improvements and reduced randomness costs. However, these tools can not verify larger components such as one, or several, masked round functions of a cipher due to complexity constraints. At this point, uniformity should be used to ease the verification process of the entire circuit.

We add that the security analysis from Chapter 5 can be used to account for non-uniformity. However, we advise extreme caution. We believe imbalances (following the definition from Daemen [34]) on an S-box level will be too high to achieve secure designs. Instead, non-uniformity can be used on a much more relaxed scale, by for example re-using some randomness several rounds apart in a cipher.

4.6 Application to AES

As an example, we provide a first-order sharing of the AES with a low randomness cost. The sharing is based on the one by De Cnudde *et al.* [43] which is a shared tower-field decomposition as proposed by Canright [24]. This S-box sharing is then placed in a changing of the guards technique as first proposed by Daemen [35].

The original sharing by De Cnudde used 11056 random bits in order to first-order secure the AES. This randomness cost has been improved by Groß *et al.* [63] to 3856 random bits. Other sharings report a similar cost, for example the multiplicative masking method by De Meyer *et al.* [47] requires 4056 random bits to secure the AES. We note that there are two other sharings of the AES which require no fresh randomness by Wegener *et al.* [101] and Sugawara [97]. However, those sharings require more shares, and as a result come with a significant area overhead.

Instead, the masking outlined below requires a total of 318 random bit, an over tenfold improvement over the sharings by Groß *et al.* and De Meyer *et al.*, while bringing no overhead compared to the design of De Cnudde *et al.*. We recall the comparison given in Section 3.4. In Table 4.1, we add our own design to this comparison by using the area cost of the design of De Cnudde *et al.* given in the work by De Meyer *et al.* [47].

Table 4.1: We compare our design with other works and provide their original area cost (using the Nangate 45nm library) along with the additional area cost to generate the randomness using Trivium from Table 3.1.

Scheme	Original Cost	Additional Cost
De Cnudde [43] *et al.*	7682GE	4644GE
Groß *et al.* [63]	7337GE	1548GE
De Meyer *et al.* [47]	6557GE	1634GE
This work	7909GE	172GE

Compared to the original design by De Cnudde *et al.*, we deliver an over 4000GE improvement. Compared to the design by De Meyer *et al.* which requires an approximate total cost of 8191GE, our design requires 8081GE. However, a bigger difference sets in when comparing round-based architectures as the cost of generating the randomness in the designs would multiply 16 times over. This means that our design can save as much as 20k GE over the other designs.

4.6.1 AES

We quickly introduce the AES cipher designed by Daemen and Rijmen [40] and standardised by NIST [82]. AES-128 consists of a 128-bit state and 128-bit key divided into 16 bytes. The cipher is composed of 10 rounds each applying an addition of a subkey, a bricklayer of S-Boxes, a `ShiftRows` operation, and a `MixColumns` operation. This is visually represented in Figure 4.3.

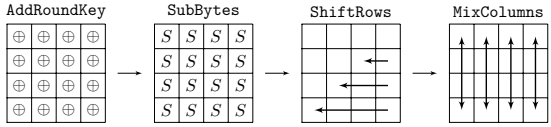


Figure 4.3: Representation of the AES.

4.6.2 Changing of the Guards With Randomness

The changing of the guards method proposed by Daemen [35] is a technique that transforms a non-complete sharing into a uniform and non-complete sharing. The technique works by embedding the sharing into a Feistel-like structure. In this paper, we slightly generalise the method by considering a first-order probing secure sharing. Such a sharing potentially requires multiple register stages and extra randomness to guarantee its security. The adapted changing of the guards method still ensures uniformity and first-order probing security while allowing the re-use of the randomness. An example of the method with two shares is shown in Figure 4.4.

We give the changing of the guards method formally.

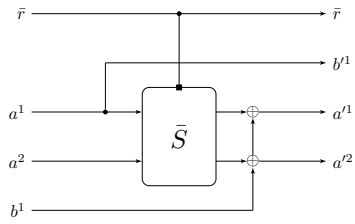


Figure 4.4: Changing of the guards method with two shares where the shared S-box $\bar{S}$ uses the randomness $\bar{r}$ in its computation (as input) to ensure it is probing secure.

Definition 4.6. The changing of the guards method applied to a shared map $\bar{S}$ given inputs $(a^1, ..., a^s)$, $(b^1, ..., b^{s-1})$, and randomness $\bar{r}$ is calculated as follows

$$\bar{r}' = \bar{r} \tag{4.1}$$

$$a'^1 = S^1(a^1, ..., a^s, \bar{r}) \oplus b^1, \quad \dots \quad , \quad a'^{s-1} = S^{s-1}(a^1, ..., a^s, \bar{r}) \oplus b^{s-1}, \tag{4.2}$$

$$a'^s = S^s(a^1, ..., a^s, \bar{r}) \oplus b^1 \oplus \ldots \oplus b^{s-1} \tag{4.3}$$

$$b'^1 = a^1, \quad \dots \quad , \quad b'^{s-1} = a^{s-1}\,. \tag{4.4}$$

In general we refer to the $(b^1, ..., b^{s-1})$ as the *guards* of the shared S-box $\bar{S}$.

We show that the changing of the guards construction with randomness retains the correctness and probing security properties from $\bar{S}$, but makes the sharing uniform.

Theorem 4.5. *The method from Definition 4.6 is correct, first-order probing secure, and uniform.*

Proof. Correctness of the construction follows from the correctness of $\bar{S}$ and the fact that each share b^i is added to two different output shares in Equations 4.2-4.3.

First-order probing security of the construction, assuming a joint uniform input, follows from the first-order probing security of $\bar{S}$ and the facts that the share b^s is not used in the construction and that each share b'^i is calculated using only one share a^i using Equations 4.2-4.3.

For the proof of uniformity, we first take an arbitrary input secret a. We show that the above construction is invertible. In other words, given the secret a and

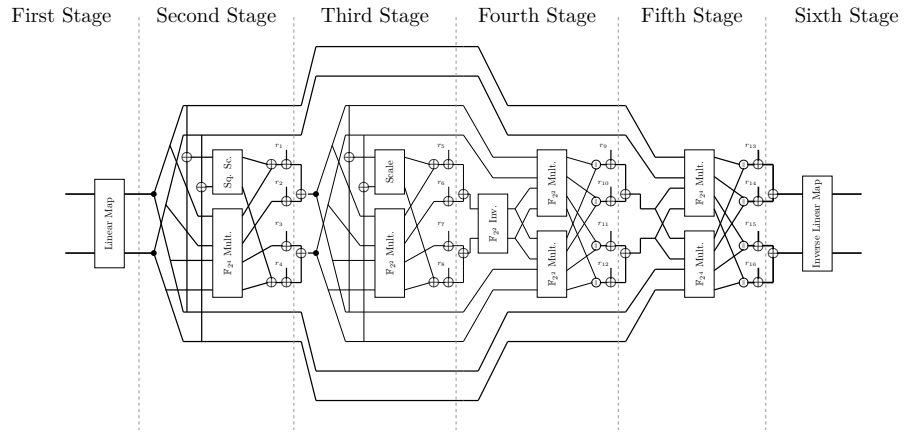


Figure 4.5: Representation of the S-box. Register stages are denoted by dashed vertical lines.

the outputs $(a'^1, ..., a'^s)$, $(b'^1, ..., b'^{s-1})$, $\bar{r}'$, we show it is possible to construct the inputs $(a^1, ..., a^s)$, $(b^1, ..., b^{s-1})$, $\bar{r}$.

Since the input secret a is given, we can construct the inputs $(a^1, ..., a^s)$ from $(b'^1, ..., b'^{s-1})$ using Equation 4.4. From $\bar{r}'$ we can evidently construct $\bar{r}$ since the two are equal (see Equation 4.1). By running $(a^1, ..., a^s)$ and $\bar{r}$ through $\bar{S}$ and XORing the output $(a'^1, ..., a'^s)$ =, we can also construct $(b^1, ..., b^s)$ (see Equations 4.2-4.3) which concludes the proof. □

Thus, the changing of the guards method allows for the transformation of any first-order probing secure sharing into a uniform one which allows the re-use of the randomness used in the S-box.

4.6.3 S-Box Sharing

We take the design by De Cnudde *et al.* [43] and place it in the changing of the guards structure detailed above. For brevity, we do not go into detail of the sharing. The sharing is depicted in Figure 4.5. In total the shared S-box requires 54 random bits. However, these random bits can be re-used over all shared S-boxes in both the AES state function as the key schedule. More information on the scheme, along with various trade-offs between area and latency can be found in the original work [53].

4.7 Lack of Higher-Order Security

In this chapter, we investigated a potential solution, namely threshold implementations, for the high randomness cost of maskings. Indeed, the technique seems to be applicable to any symmetric primitive thanks to the changing of the guards technique given in Section 4.6.2. These threshold implementations also come at no significant extra cost if the masking is carefully investigated as shown in Section 4.6 in the application to the AES. However, there is one limitation to the work by Nikova *et al.* which merits further study. Threshold implementations only provide first-order probing security and it has been a long-standing open problem how to secure them against higher-order attacks.

Chapter 5

Higher-Order Threshold Implementations

In this chapter, we introduce a way to verify the security of a masked symmetric primitive in the bounded-query probing model from Section 2.6.3. The analysis method allows to step away from information-theoretic maskings. We apply the method to a masking of the LED cipher and observe we can remove a large part of the randomness cost of higher-order maskings.

Everything in this chapter is written by myself and Tim Beyne. The writing in this chapter is based on a paper with Tim Beyne and Zhenda Zhang [12] and is published at Asiacrypt. This chapter was made in equal collaboration between myself, providing knowledge from the field of masking, and Tim Beyne, providing knowledge from the field of cryptanalysis.

5.1 The Historical Context

The concept of *higher-order threshold implementations* is not new. In 2014, Bilgin *et al.* [15] published a paper titled "Higher-Order Threshold Implementations", the goal of the paper being the evident. However, only a year later a note by Reparaz was made public that the method described in the paper was insecure [87]. Essentially, the work by Bilgin *et al.* protects against higher-order univariate attacks, attacks where information from within the same cycle is used, but it fails to protect against multivariate attacks, attacks combining information from multiple time samples in the trace. In that

same year, a solution was proposed by Reparaz *et al.* [88] to secure against second and third-order attacks and, again, in 2018 Moos *et al.* found flaws in the third-order version of that scheme [77]. Even besides the third-order flaw, the work by Reparaz *et al.* requires (a lot of) fresh randomness in order to guarantee this security. Given the main research topic of this thesis, we believe such a solution is not in the spirit of the original work on threshold implementations.

As such, we wish to propose a new analysis method. The method allows to research the security of maskings which are not information-theoretic secure, but reach some security bound in the bounded-query model from Section 2.6.3. Due to the analysis method, we can build higher-order secure maskings which are more efficient than those built using other analysis methods, like the non-interference model from Section 2.6.2. In fact, similar to threshold implementations versus first-order non-interferent maskings, as discussed in Section 4.4, these new higher-order maskings include those that need no fresh randomness in their computation. We thus show how to solve the limitation outlined in Section 4.7 without removing the benefits.

5.2 Fourier Analysis

Before we head to the analysis method, we first provide some information on Fourier analysis. The relation between linear cryptanalysis and the Fourier transformation on vector spaces over $\mathbb{F}_2$ is well-established [9, 25, 37]. This section introduces the necessary notation for two important concepts that will be used throughout this chapter. The first is the Fourier transformation of a probability distribution, or more generally any complex-valued function, on a vector space V over $\mathbb{F}_2$. The second is the notion of the correlation matrix of a function $F : V \to V$, the coordinates of which coincide with the correlations of linear approximations over F.

The definitions below are formulated for functions on an arbitrary subspace $V \subseteq \mathbb{F}_2^n$, as opposed to functions on $\mathbb{F}_2^n$ itself – as is commonly the case. Since any vector space over $\mathbb{F}_2$ is isomorphic to $\mathbb{F}_2^n$ for some n, this generalisation is mostly a matter of notation. Nevertheless, this extended notation will be very beneficial later on in this work.

Recall that the orthogonal complement of a subspace V of $\mathbb{F}_2^n$ is the vector space $V^\perp = \{x \in \mathbb{F}_2^n \mid \forall v \in V : v^\top x = 0\}$. The quotient space $\mathbb{F}_2^n / V^\perp$ is by definition the vector space of cosets of $V^\perp$. For convenience, an element $x + V^\perp \in \mathbb{F}_2^n / V^\perp$ will simply be denoted by x. For $x \in \mathbb{F}_2^n / V^\perp$ and $v \in V$, the expression $x^\top v$ is well-defined. Consequently, the following definition is proper.

Definition 5.1. Let $V \subseteq \mathbb{F}_2^n$ be a vector space and $f : V \to \mathbb{C}$ be a complex-valued function on V. The Fourier transformation of f is a function $\widehat{f} : \mathbb{F}_2^n/V^\perp \to \mathbb{C}$ defined by

$$\widehat{f}(u) = \sum_{x \in V} (-1)^{u^\top x} f(x),$$

where we write u for $u + V^\perp$. Equivalently, $\widehat{f}$ is the representation of f in the basis of functions $x \mapsto (-1)^{u^\top x}$ for $u \in \mathbb{F}_2^n/V^\perp$.

Unlike in standard treatments of the Fourier transformation on finite abelian groups [54, 92], Definition 5.1 introduces $\widehat{f}$ as a function on $\mathbb{F}_2^n/V^\perp$ rather than on the Pontryagin dual group $\widehat{V}$. Throughout this document, the isomorphism $\widehat{V} \cong \mathbb{F}_2^n/V^\perp$ is used to simplify notation. Other choices are possible, but this one is the most convenient.

As one often encounters transformations of random variables, it is convenient to encode the action of a function $F : V \to V$ on probability distributions as a linear operator. The coordinate representation of this operator with respect to the standard basis $\{\delta_x\}_{x \in V}$ may be called the *transition matrix* of F. Following [8, 9], the correlation matrix of F is then the same operator expressed with respect to the Fourier basis. Note that correlation matrices for use in cryptanalysis were first introduced by Daemen *et al.* [37], where their definition is given in terms of concepts from linear cryptanalysis.

Definition 5.2 (Transition matrix)**.** Let V be a set and $F : V \to V$ a function. The transition matrix T^F of F is a real $|V| \times |V|$ matrix such that, in the standard basis, if $\mathbf{x}$ has probability mass function $p_{\mathbf{x}} : V \to [0, 1]$, then $F(\mathbf{x})$ has probability mass function $T^F p_{\mathbf{x}}$. Equivalently, indexing the coordinates of T^F by elements of V, we have $T^F_{y,x} = \delta_{y,F(x)}$.

Definition 5.3 (Correlation matrix)**.** Let $V \subseteq \mathbb{F}_2^n$ be an $\mathbb{F}_2$-vector space and $F : V \to V$ a function. The correlation matrix C^F of F is the coordinate representation of the linear map defined by T^F with respect to the basis of character functions $x \mapsto (-1)^{u^\top x}$ for $u \in \mathbb{F}_2^n/V^\perp$. Equivalently, indexing the coordinates of C^F by elements of $\mathbb{F}_2^n/V^\perp$, we have

$$C^F_{v,u} = \frac{1}{|V|} \sum_{x \in V} (-1)^{u^\top x + v^\top F(x)}.$$

The relation between Definition 5.3 and linear cryptanalysis is as follows: the coordinate $C^F_{v,u}$ is equal to the correlation of a linear approximation over F with input mask u and output mask v. That is, $C^F_{v,u} = 2 \Pr[v^\top F(\mathbf{x}) = u^\top \mathbf{x}] - 1$ for $\mathbf{x}$ uniform random on V.

5.3 Bound on the Advantage

This section highlights the most important theorem from the chapter. Namely, we give an upper bound on the bounded-query probing adversary from the Fourier distribution of the probed random variables.

The connection starts with a lemma described below. The connection between the entropy of a probability distribution and its Fourier transformation as defined above. The proof of the lemma is found in our work [12].

Lemma 5.1. *Let* $\mathbf{x}$ *be a random variable on* $\mathbb{F}_2^m$ *with probability distribution* $p_{\mathbf{x}}$*. It holds that*

$$m - \mathsf{H}(\mathbf{x}) \leq \|\widehat{p}_{\mathbf{x}} - \delta_0\|_2^2 / \log 2,$$

where $\mathsf{H}(\mathbf{x})$ *denotes the information entropy of* $\mathbf{x}$ *with respect to the binary logarithm.*

The inequality in the above lemma is sharp when $\|\widehat{p}_{\mathbf{x}} - \delta_0\|_2^2 \ll 1$. This is the case when $\widehat{p}_{\mathbf{x}}$ has a small support. Typically, a probing adversary can observe only a small portion of the state of a symmetric primitive making the above lemma sharp.

We then connect the advantage of a bounded-query probing adversary to the entropy of the observed probability distribution. Using the above lemma, this connection is expanded to bound the advantage of the adversary by the Fourier transformation of the distribution. Before viewing the main theorem, we briefly consider the content of its statement. The theorem considers two cases, either all probed wire values closely resemble uniform randomness or they reveal nothing about the secret input. The second case arrives from the regular probing model by Ishai *et al.* [69]. However, it is the first case which provides the usefulness of the result. Namely, we allow for 'bad' probe values. These are values that might leak information about the secret, but which nevertheless cannot be distinguished from uniform random values unless a very large number of probing queries is made. In practice, the 'bad' values will be shares of the state resulting from probes placed far apart (*i.e.* separated by many rounds). The 'good' values then correspond to probes that are placed in nearby locations, such as within an S-box. Again, the proof of the following theorem can be found in our work [12].

Theorem 5.2. *Let* $\mathcal{A}$ *be a* t*-threshold-probing adversary for a circuit* C*. Assume that for every query made by* $\mathcal{A}$ *on the oracle* $\mathcal{O}^b$*, there exists a partitioning (depending only on the probe positions) of the resulting wire values into two random variables* $\mathbf{x}$ *('good') and* $\mathbf{y}$ *('bad') such that*

1. *The conditional probability distribution* $p_{\mathbf{y}|\mathbf{x}}$ *satisfies* $\mathbb{E}_{\mathbf{x}}\|\widehat{p}_{\mathbf{y}|\mathbf{x}} - \delta_0\|_2^2 \leq \varepsilon$*,*

2. *Any t-threshold-probing adversary for the same circuit C and making the same oracle queries as $\mathcal{A}$, but which only receives the 'good' wire values (*i.e. *corresponding to $\mathbf{x}$) for each query, has advantage zero.*

The advantage of $\mathcal{A}$ can be upper bounded as

$$\mathrm{Adv}_{t\text{-}\mathsf{thr}}(\mathcal{A}) \leq \sqrt{2q\,\varepsilon}\,,$$

where q is the number of queries to the oracle $\mathcal{O}^b$.

The above theorem thus allows a security analysis in the bounded-query probing model based on the Fourier transform of the probed distribution. This Fourier transform is calculated using linear cryptanalytic methods, which have been investigated for more than thirty years. As a result, we can easily give bounds on the adversary of a particular masking. We will see this applied to an example in Section 5.6.

5.4 Fourier Analysis of Shared Functions

In the previous section, we proposed Theorem 5.2 which details an upper bound on the advantage of probing adversaries in terms of the Fourier coefficients of the probability distribution of observed wire values. In this section, we clarify the use of the theorem. More specifically, we show the link with the linear cryptanalysis of shared functions.

5.4.1 Restrictions of Shared Functions

Remark that all probability distributions referred to in Theorem 5.2 are with respect to a fixed value of the secret inputs. Consequently, it is clear that the relevant Fourier coefficients can not be directly related to the Walsh-Hadamard coefficients (or equivalently, the correlation matrix) of the shared function itself. Instead, the relevant properties are those of restrictions of the shared function to sets of all valid sharings of a specific secret. Below, we argue that these restrictions are indeed well-defined and that they come with a natural notion of linear cryptanalysis.

Recall from Section 2.3 that masking is based on linear secret sharing. In general, any $\mathbb{F}_2$-linear secret sharing scheme can be thought of as an algorithm that maps a secret $x \in \mathbb{F}_2^n$ to a random element of a corresponding coset of a vector space $\mathbb{V} \subset \mathbb{F}_2^\ell$. The vector space $\mathbb{V}$ consists of all possible sharings of

$0 \in \mathbb{F}_2^n$. Let $\rho : \mathbb{F}_2^n \to \mathbb{F}_2^\ell$ be a map that sends secrets to their corresponding coset representative.

Example Consider the Boolean masking of a secret $x \in \mathbb{F}_2$ shared as $(x^1, \ldots, x^\ell)$, then $\mathbb{V}$ corresponds to the parity bit code

$$\mathbb{V} = \{(x^1, \ldots, x^\ell) \in \mathbb{F}_2^\ell \mid \sum_{i=1}^{\ell} x^i = 0\}.$$

Furthermore, one possible choice of ρ is $\rho(x) = (x, 0, \ldots, 0)$.

Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be any function. Recall from Definition 4.2 that a function $\bar{F} : \mathbb{F}_2^\ell \to \mathbb{F}_2^\ell$ is said to be a correct sharing of F if, for all $x \in \mathbb{F}_2^n$,

$$\bar{F}(\rho(x) + \mathbb{V}) \subseteq \rho(F(x)) + \mathbb{V}. \tag{5.1}$$

If $\bar{F}$ is a uniform sharing, then the above inclusion is in fact an equality. For convenience, let $\mathbb{V}_a = a + \mathbb{V}$. Due to (5.1), the restriction of $\bar{F}$ to $\mathbb{V}_a$ is a well-defined function $\mathbb{V}_a \to \mathbb{V}_b$ whenever $a = \rho(x)$ and $b = \rho(F(x))$ for some $x \in \mathbb{F}_2^n$. By slight abuse of notation, the same notation will be used for $\bar{F}$ and for its restrictions.

Any random variable $\mathbf{x}$ on $\mathbb{V}_a$ has a corresponding probability mass function $p_{\mathbf{x}} : \mathbb{V}_a \to [0, 1]$. Since $\mathbb{V}$ is a vector space, the Fourier transformation $\widehat{p}_{a+\mathbf{x}}$ of $p_{a+\mathbf{x}}$ is well-defined (see Definition 5.1). In addition, for any restriction $\bar{F} : \mathbb{V}_a \to \mathbb{V}_b$, the correlation matrix of $x \mapsto \bar{F}(a + x) + b$ is well defined by Definition 5.3. For convenience, we introduce the following definition (with slight abuse of notation). Note that it does not depend on the choice of the coset representatives a and b.

Definition 5.4. For $\mathbb{V} \subseteq \mathbb{F}_2^\ell$, let $\bar{F} : \mathbb{V}_a \to \mathbb{V}_b$ be a well-defined restriction of a shared function. Let $\bar{F}'(x) = \bar{F}(x + a) + b$. The absolute correlation matrix $|C^F|$ of $\bar{F}$ is defined as the matrix with coordinates

$$|C^{\bar{F}}_{v,u}| = |C^{\bar{F}'}_{v,u}|,$$

for $u, v \in \mathbb{F}_2^\ell / \mathbb{V}^\perp$.

5.4.2 Correlations between Probed Values

We have seen that we can define the correlation matrix of a sharing with a fixed secret. We now apply this to Theorem 5.2. In this theorem, it was shown that the advantage of a probing adversary against the circuit can be upper bounded in terms of $\|\widehat{p}_{\mathbf{z}} - \delta_0\|_2$ where $p_{\mathbf{z}}$ is the probability distribution of any

measured set of 'bad' wire values, possibly conditioned on several 'good' wire values. Note that the conditioning on 'good' values simply corresponds to fixing some variables in the circuit to constants before applying the results below. This section provides the essential link between $\widehat{p}_{\mathbf{z}}$ and the linear cryptanalysis of the shared circuit that will enable us to upper bound the quantity $\|\widehat{p}_{\mathbf{z}} - \delta_0\|_2$.

For simplicity, from this point on, we only consider second-order probing adversaries. To obtain the desired link with linear cryptanalysis, it will be shown that the coordinates of $\widehat{p}_{\mathbf{z}}$ are entries of the correlation matrix of the state-transformation between the specified probe locations. This is illustrated in Figure 5.1.

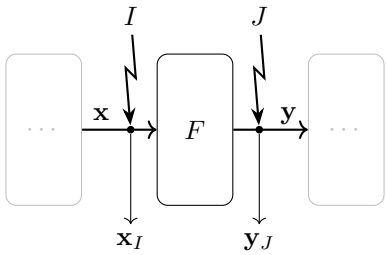


Figure 5.1: Two probes giving the observation $\mathbf{z} = (\mathbf{x}_I, \mathbf{y}_J)$.

The main result is stated in Theorem 5.4. To obtain it, the following property of correlation matrices will be used.

Lemma 5.3. *Let $\mathbb{V} \subset \mathbb{F}_2^\ell$ be a set of correct sharings and $L : \mathbb{V} \to \mathbb{F}_2^m$ a linear map. If $\mathbf{x}$ is a random variable with probability distribution $p_{\mathbf{x}}$, then it holds that*

$$\widehat{p}_{L\mathbf{x}}(u) = \widehat{p}_{\mathbf{x}}(L^\top u),$$

where we write $L^\top u = L^\top u + \mathbb{V}^\perp$ as usual.

For an index set $I = \{i_1, \ldots, i_m\}$, we denote the restriction of $x \in \mathbb{V}$ to I by $x_I = (x_{i_1}, \ldots, x_{i_m}) \in \mathbb{F}_2^{|I|}$. Note that the maps $x \mapsto x_I$ and its restriction to $\mathbb{V}$ are linear.

Theorem 5.4. *Let $F : \mathbb{V}_a \to \mathbb{V}_b$ be a function with $\mathbb{V} \subset \mathbb{F}_2^\ell$ and $I, J \subset \{1, \ldots, \ell\}$. For $\mathbf{x}$ uniform random on $\mathbb{V}_a$ and $\mathbf{y} = F(\mathbf{x})$, let $\mathbf{z} = (\mathbf{x}_I, \mathbf{y}_J)$. The Fourier transformation of the probability mass function of $\mathbf{z}$ then satisfies*

$$|\widehat{p}_{\mathbf{z}}(u, v)| = |C^F_{\widetilde{v}, \widetilde{u}}|,$$

where $\widetilde{u}, \widetilde{v} \in \mathbb{F}_2^\ell / \mathbb{V}^\perp$ are such that $\widetilde{u}_I = u$, $\widetilde{u}_{[\ell]\setminus I} = 0$, $\widetilde{v}_J = v$ and $\widetilde{v}_{[\ell]\setminus J} = 0$.

The proof of the above theorem and lemma are found in our work [12].

Theorem 5.4 relates the linear approximations of F to $\widehat{p}_{\mathbf{z}}(u)$ and hence provides a method to upper bound $\|\widehat{p}_{\mathbf{z}} - \delta_0\|_2$ based on linear cryptanalysis. However, it should be noted that the result relates to linear cryptanalysis with respect to $\mathbb{V}$ rather than $\mathbb{F}_2^\ell$. The differences are mostly minor, but there is a subtle difference in relation to the important notion of 'activity'. In standard linear cryptanalysis, an S-box is said to be *active* if its output mask is nonzero. Conversely, an S-box is *inactive* if its output mask is zero. The same definition applies for linear cryptanalysis with respect to $\mathbb{V}$, but one must take into account that the mask is now an element of the quotient space $\mathbb{F}_2^\ell/\mathbb{V}^\perp$. In particular, if the mask corresponding to the shares of a particular bit can be represented by an all-one vector $(1, 1, \ldots, 1)^\top$, it may be equivalently represented by the zero vector. It is still true that a valid linear approximation for a permutation must have either both input masks equivalent to zero or neither equivalent to zero. More generally, this condition is ensured by any uniform sharing.

Finally, note that Theorem 5.4 assumes that all intermediate states of the shared implementation are uniformly distributed on a coset of $\mathbb{V}$. This condition is guaranteed by the uniformity property of threshold implementations. In fact, it corresponds to the fact that the approximation with – up to equivalence – an all-zero input mask, must also have an all-zero output mask in order to have nonzero correlation. In particular, this is achieved if all shared functions are permutations.

5.5 Higher-Order Non-Completeness

The above theorems are useful to analyse the security of a masking where the probes are placed on different rounds in a cipher. However, we still need to tackle how to secure one round of the masked cipher against a higher-order probing adversary without using any fresh randomness. Luckily, this question was already solved some years ago. Recall the work by Bilgin *et al.* [15] on higher-order threshold implementations. This work was shown to be insecure, however, that attack essentially only works when several masked functions are composed together. Instead, Bilgin *et al.* introduces a notion of higher-order non-completeness which locally secures a masked function without the use of fresh randomness.

Definition 5.5 (d^{th}-order non-completeness)**.** A shared function $\bar{F}(\bar{x})$ is d^{th}-order non-complete if every set of d shares of $\bar{F}$ jointly operate on at most $s_x - 1$ input shares.

To achieve the above notion, shared functions operating on more shares should be made. In general, an algebraic degree t function can be shared in a dth-order non-complete sense using at least $td + 1$ shares.

5.6 Application to LED

The above sections show an analysis method for masking schemes. In essence, the idea is to show that the probed values either contain no information on any secret (regular probing security), or we show that the values are close to uniform random using Fourier analysis. This Fourier analysis comes down to investigating the security of the masking from the view of linear cryptanalysis. As such, the security analysis should appear quite regular to symmetric-key researchers.

We apply the analysis technique on a concrete example, namely a second-order masking of the LED cipher by Guo *et al.* [65]. In total, this masking requires 664 bits of randomness, *i.e.* 24 bits more than what is needed to share the plaintext and key.

5.6.1 Description of LED

LED is a 64-bit block cipher where the state is divided into 16 four-bit cells. The variant considered here has a 128-bit master key, from which subkeys are derived using a nibble-wise permutation. The cipher consists of 12 steps, each comprising four rounds. The step function is shown in Figure 5.2. For further details, we refer the reader to the work of Guo *et al.* [65].

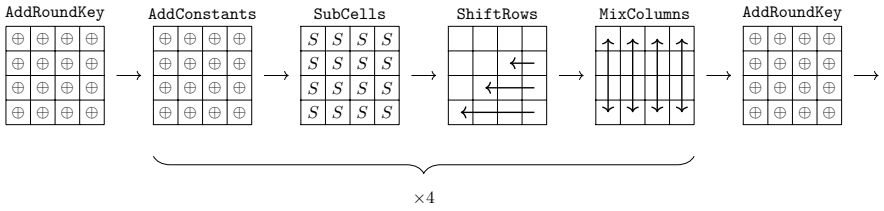


Figure 5.2: The step function of LED.

5.6.2 Sharing Second-Order LED

This section constructs a sharing of the LED cipher. Figure 5.3 gives an overview of the shared round function.

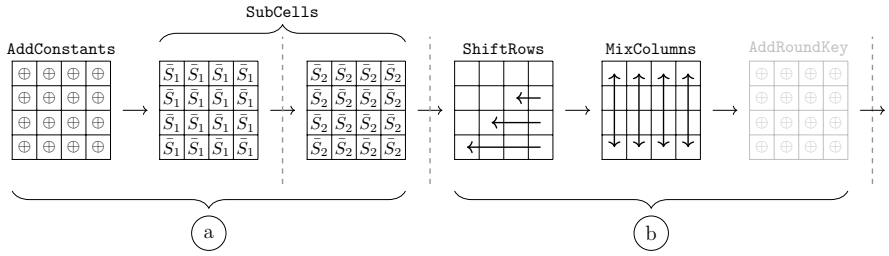


Figure 5.3: One round of masked LED. The locations of the registers are indicated by dashed lines. The round key addition is depicted in grey to show that it only happens every four rounds.

Masking State and Key For the sharing of LED we use classical Boolean masking. The 64-bit state is shared using seven shares per bit, requiring 384 random bits. The 128-bit key is shared using three shares, which costs 256 random bits.

Sharing Affine Components The masking of LED's linear components such as `ShiftRows`, `MixColumns`, and the key schedule are simply done share-wise. Constants are added to the first share of the concerning variable. The key addition is done by adding the key shares to the first three shares of the state.

Sharing the S-Box LED uses the PRESENT S-box. Following the work by Kutzner *et al.* [74], this S-box can be decomposed into two quadratic maps $S_1 = G \circ C$ and $S_2 = B \circ G$ where B and C are affine. More specifically, we decompose as follows

$$S(x, y, z, w) = B'(G(G(C'(x, y, z, w) + d)) + e)\,.$$

In the above, the nonlinear function $G(x, y, z, w)$ is given as

$$G_1 = x + yz + yw \qquad G_2 = w + xy \qquad G_3 = y \qquad G_4 = z + yw\,,$$

the linear transformations as

$$B' = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}, C' = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix},$$

and the constants as

$$d = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}, e = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}.$$

The sharing of the S-box is constructed from the sharing of G. For each share $i \in \{1, ..., 7\}$, the permutation $G(x, y, z, w)$ is shared as

$$G_1^i = x^i + y^i z^i + y^i z^{i+1} + y^{i+1} z^i + y^i z^{i+3} + y^{i+3} z^i + y^{i+1} z^{i+3} + y^{i+3} z^{i+1}$$
$$+ y^i w^i + y^i w^{i+1} + y^{i+1} w^i + y^i w^{i+3} + y^{i+3} w^i + y^{i+1} w^{i+3} + y^{i+3} w^{i+1},$$

$$G_2^i = w^i + x^i y^i + x^i y^{i+1} + x^{i+1} y^i + x^i y^{i+3} + x^{i+3} y^i + x^{i+1} y^{i+3} + x^{i+3} y^{i+1},$$

$$G_3^i = y^i,$$

$$G_4^i = z^i + y^i w^i + y^i w^{i+1} + y^{i+1} w^i + y^i w^{i+3} + y^{i+3} w^i + y^{i+1} w^{i+3} + y^{i+3} w^{i+1},$$

where the convention is used that superscripts wrap around at seven.

The above sharing has been verified to be uniform and second-order non-complete. In between the two G functions, a layer of randomness consisting of zero sharings is added. This randomness is re-used in every S-box call and consists of 24 bits in total.

Security In Sections 5.6.3 to 5.6.6 below, the following concrete security claim will be established.

Security Claim 1. For the masked LED described in this section, the following bound on the advantage of the adversary (assuming piling-up) in the probing model is claimed:

$$\mathrm{Adv}_{\text{2-thr}}(\mathcal{A}) \leq \sqrt{\frac{q}{2^{121}}}.$$

5.6.3 Probing Security of One Round

This section establishes the second-order probing security of one round of the masked LED, such that all wire values corresponding to such probing queries

can be labelled as 'good'. Recall that, since each layer of the masked cipher is uniformly shared, the input distribution to the round is uniform. To establish the probing-security claim, it suffices to consider all possible probe positions. If both probes are placed in the same layer, the claim follows directly from the second-order non-completeness of each function.

When both probes are placed in part (a) in Figure 5.3, the only nontrivial new case corresponds to placing one probe in $\bar{S}_1$ and one in $\bar{S}_2$. Due to the refreshing layer $\bar{R}$, the input to $\bar{S}_2$ is uniformly random even if $\bar{S}_1$ is probed. Since $\bar{S}_2$ is second-order non-complete, placing the second probe in $\bar{S}_2$ then reveals no information about the secret.

If one probe is placed in part (a) and another in part (b), then the second probe reveals at most a single share (the same) of each variable by the linearity of part (b). Due to a consistent choice of the covering scheme used for non-completeness, the previous arguments are not limited to the bit-level. Consequently, the analysis is the same as for the case with two probes in part (a).

Every four rounds, a round key is also added to the state. The effect of the key-schedule and key addition is discussed in Section 5.6.6.

5.6.4 Probing Nearby Rounds: Zero Correlation

This section shows the distribution of any pair of measurements from probes which are at most three rounds apart almost always conforms to one of two cases: either the observations are uniformly distributed, or they do not reveal anything about the secret. To prove the uniformity claim of our observations, we rely on techniques from *zero-correlation linear cryptanalysis*. The latter case, *i.e.* independence of the secret for possibly non-uniform observations, was discussed in the previous section. For these cases, the advantage of the adversary is zero as specified in the proof of Theorem 5.2. All other cases will be considered in Section 5.6.5.

The argument consists of an analysis of all possible probe placements. As noted above, the analysis in this section is restricted to probes that are at most three rounds apart. This results in the following cases:

Rounds i and $i+1$. If the adversary probes in part (a) of round i, then the MDS matrix ensures that a full column of the state will be active at the input of round $i+1$. A measurement in part (a) of round $i+1$ can activate shares from at most one cell of the state such that the corresponding approximations have correlation zero. Similarly, due to the shift rows operation, by probing in part (b) of round $i+1$, the adversary can never

activate all cells of a single column at the input of round $i+1$. Hence, approximations with nonzero correlation can only be obtained by probing in part (b) of round i. However, in this case only a single share of each bit is learned, such that a second probe in part (a) or (b) of round $i+1$ reveals nothing about the secret by the same argument for the case where both probes are placed in round i.

Rounds i and $i+2$. If either part (a) or (b) of round i are probed, this results (up to symmetry) in one of the four activity patterns shown in Figure 5.4 for rounds $i+1$ and on. By probing anywhere in round $i+2$, the adversary can clearly activate at most four cells at the input of this round. In cases (1)–(3) in Figure 5.4, at least eight S-boxes are active at the input of round $i+2$ such that the correlation of such approximations is zero. In the remaining case, *i.e.* activity pattern (4), only a single column of the state is active at the input of round $i+2$. However, by probing in part (a) of round $i+2$, only a single cell can be activated. Probing part (b) allows activating four cells but never from the same column due to the shift rows step.

Rounds i and $i+3$. It is easy to see that activity patterns (2)-(4) in Figure 5.4 lead to correlation zero since at least eight S-boxes are then active at the input of round $i+3$. Indeed, if the second probe is placed anywhere in round $i+3$, at most four cells of the state can be activated. For pattern (1) in Figure 5.4, the correlation may be nonzero and will be bounded in Section 5.6.5.

The above case analysis shows that, when the probes are placed in nearby rounds, perfect security is obtained. The only remaining cases are probes in rounds i and $i+r$ for $r > 4$ and the activity pattern (1) in Figure 5.4 when probes are placed in rounds i and $i+3$. These cases are analysed in Section 5.6.5.

5.6.5 Five Rounds or More: Low Correlation

As discussed in Section 5.6.4, if the probes are placed in rounds that are far apart, the observed values are usually not uniformly distributed. Nevertheless, it is possible to show that they will be nearly uniform in the sense that all nontrivial coordinates of the Fourier transform of their probability distribution are small. To show this, we rely on standard techniques from linear cryptanalysis: we bound the correlation of all linear trails whose activity pattern is compatible with the probe positions.

The analysis in this section relies on the *piling-up approximation* [76, 98], *i.e.* we use upper bounds on the correlation of trails as an approximation for upper

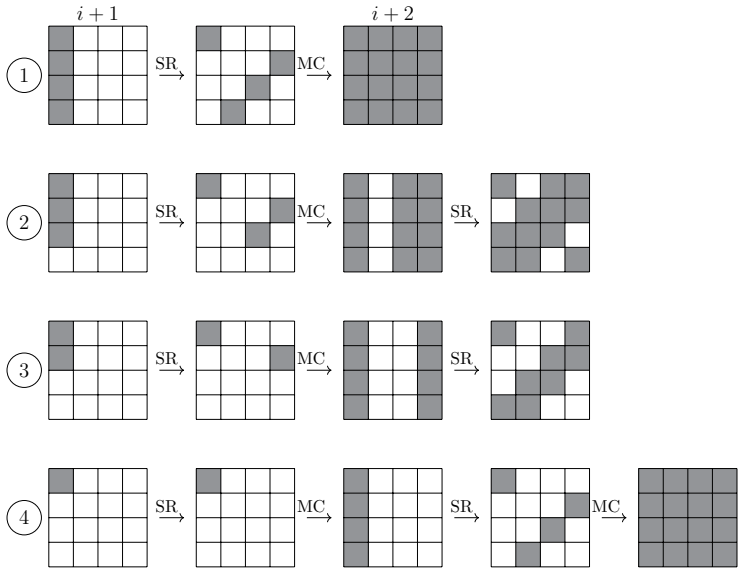


Figure 5.4: Activity patterns for masked LED, corresponding (up to symmetry) the four possible patterns created by a probe placed in round i. SR is short for `ShiftRows` and MC for `MixColumns`. White cells are inactive and cells in grey are active.

bounds on the correlation of linear approximations. This heuristic is widely used in symmetric-key cryptanalysis, and is an integral part of the security argument of essentially all contemporary symmetric-key primitives. In addition, as detailed in Section 5.6.6, our correlation upper bounds need not hold for all key and refreshing variables but only in the average over the unobserved variables. Finally, any adversary that can distinguish the probed wire values from uniform randomness gives rise to a linear distinguisher. Consequently, we believe it is reasonable to apply the piling-up heuristic in this setting.

To upper bound absolute trail correlations, we rely on the standard wide-trail argument [39]. Specifically, the fact that any linear trail over four rounds of (shared) LED activates at least 25 S-boxes will be used. Additionally, an upper bound on the correlation of the best linear approximations over the shared S-box from Section 5.6.2 is required. Since the shared S-box is quite large, a direct calculation of its nonlinearity is nontrivial. Instead, the following lemma for quadratic Boolean functions can be used. A slight restatement of this result can be found in the book by Carlet [25, Chapter 6].

Lemma 5.5 (Proposition 16 [25]). *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a quadratic Boolean function. Denote the rank of its symplectic form by r. That is, $r = \mathrm{rank}(S)$ where $S \in \mathbb{F}_2^{n\times n}$ is the symmetric matrix for which $y^\top S\, x = f(x+y)+f(x)+f(y)$. Then*

$$\frac{1}{2^n}\Big|\sum_{x\in\mathbb{F}_2^n}(-1)^{f(x)}\Big| \le 2^{-r/2}.$$

The following two lemmas were verified in the paper [12].

Lemma 5.6. *Let $\bar{G} : \mathbb{V}_a \to \mathbb{V}_b$ be any restriction of the sharing of G defined in Section 5.6.2. Denote its absolute correlation matrix by $|C^{\bar{G}}|$. For any $u, v \in \mathbb{F}_2^\ell/\mathbb{V}^\perp$ such that $u_i^j \neq 0$ for some $i \neq 3$, it holds that $\left|C^{\bar{G}}_{u,v}\right| \le 2^{-3}$.*

Theorem 5.7. *Let $\bar{S} = \bar{S}_2 \circ \bar{S}_1 : \mathbb{V}_{a_1} \to \mathbb{V}_{a_3}$ be the sharing of $S = S_2 \circ S_1$ defined in Section 5.6.2. Denote the absolute correlation matrix of $\bar{S}_i : \mathbb{V}_{a_i} \to \mathbb{V}_{a_{i+1}}$ by $|C^{\bar{S}_i}|$. For any $u, v \in \mathbb{F}_2^\ell/\mathbb{V}^\perp$ not both equal to zero and for all $w \in \mathbb{F}_2^\ell/\mathbb{V}^\perp$, it holds that $\left|C^{\bar{S}_2}_{u,w}\right| \left|C^{\bar{S}_1}_{w,v}\right| \le 2^{-3}$.*

Experimentally, we find that the piling-up approximation gives the correct upper bound 2^{-3} for the maximum absolute correlation of the shared S-box. Due to resource constraints, the experiment was limited to the verification for one choice of randomness constants.

For probes placed in rounds i and $i + r$ with $r \ge 4$, the relevant linear trails all have at least 25 active S-boxes. This is a consequence of the *wide-trail design strategy* and can be derived in exactly the same way as for the AES [39]. Hence, by Theorem 5.7, the correlations of these trails are bounded by 2^{-75}. By Theorem 5.4, it then follows that the 2-norm of the nontrivial Fourier coefficients of the observed bits $\mathbf{z}$ can be upper bounded by

$$\varepsilon := \|\widehat{p}_{\mathbf{z}} - \delta_0\|_2^2 \le |\mathrm{supp}\,\widehat{p}_{\mathbf{z}}|\,\|\widehat{p}_{\mathbf{z}} - \delta_0\|_\infty^2 \le 2^{22}\,2^{-150} = 2^{-128},$$

where we have used the inequality $|\mathrm{supp}\,\widehat{p}_{\mathbf{z}}| \le 2^{22}$, which follows from the fact that the observed value $\mathbf{z}$ consists of at most 22 bits in the glitch-extended probing model: if an output coordinate of $\bar{G}$ is read, at most 10 shares are learned; if an output of the shared linear layer is probed, at most 11 shares are observed. The latter number of shares is due to the fact that LED's MDS matrix has at least five zeros per row when viewed over $\mathbb{F}_2$. Note that, in practice, the upper bound above is not likely to be tight, because it is unlikely that a glitch will reveal the exact value of all 11 bits in a single measurement.

The only remaining case is when the adversary probes in rounds i and $i + 3$, assuming the activity pattern in case ① from Figure 5.4. In this case, only 24

S-boxes are active. Furthermore, we again have $|\mathrm{supp}\,\widehat{p}_{\mathbf{z}}| \leq 2^{22}$. Hence,

$$\varepsilon := \|\widehat{p}_{\mathbf{z}} - \delta_0\|_2^2 \leq 2^{22}\, 2^{-144} = 2^{-122}.$$

Note that a more careful analysis results in slightly improved bounds. Nevertheless, since we believe the bound on ε is sufficiently small for all practical purposes, we avoid such an analysis and opt for simplicity instead.

5.6.6 Influence of the Key-Schedule

The arguments in Sections 5.6.4 and 5.6.5 directly establish the security of our proposed masked LED design against an adversary which does not look at shares of the key or the bits which are added in the refreshing layer. Indeed, for such an adversary, we may mark all wire values for queries with probe positions considered in Section 5.6.4 as 'good' and all others (considered in Section 5.6.5) as 'bad'. Theorem 5.2 then provides the desired upper bound. However, showing the conventional security where all wires in the circuit can be probed requires a slightly more careful choice of 'good' and 'bad' wire values.

Fortunately, the LED key-schedule consists only of bit-permutations. Hence, its sharing is perfectly secure against second-order threshold-probing adversaries. The same holds for the random bits used in the refreshing layer. Hence, Theorem 5.2 can be applied with the following labelling of wire values:

Probes discussed in § 5.6.3–5.6.4. For all these probe positions, all wire values can be considered as 'good'. This includes any key bits (and additional randomness in the refreshing layer) that might be observed by the adversary. Indeed, even with glitch-extended probes, the adversary can observe at most two shares of each key bit.

Probes discussed in § 5.6.5. For these probe positions, all wire values corresponding to state shares should be marked as 'bad'; shares of the key (or additional randomness used in the refreshing layer) are labelled 'good'. The arguments in Section 5.6.5 then apply directly.

At least one probe in the key-schedule. In this case, all wire values may be considered 'good'. Indeed, recall that any non-complete subset of state bits at a particular layer is uniformly distributed and the adversary observes at most two shares of each key bit.

For the upper bound ε, the values derived in Section 5.6.5 may be used directly – as the analysis of the trails there is valid for any choice of the key. Note that the latter assumption is stronger than necessary; it suffices to assume that the

bounds derived in Section 5.6.5 are valid in the average over all unobserved randomness and key variables.

5.7 The Remaining Engineering Challenges

In this chapter, we gave an analysis method for the bounded-query model. With this method, we are capable of analysing the security of an entire masking as opposed to the non-interference model where verification is done by analysing the security of sub-circuits. The novel analysis method circumvents the limitation on the use of randomness described in Chapter 3 and allows for the security of higher-order threshold implementations solving the limitation described in Chapter 4.

The analysis technique is still quite limited. We have shown an example of a masked LED using seven shares. However, state-of-the-art maskings typically only use three shares to achieve second-order security (unless a very favourable trade-off for latency is possible). In other words, while the masking of the LED uses almost minimal randomness, it still has a significant overhead compared to other designs. The analysis method is also complicated to do by hand for several primitives. Finally, low-randomness maskings seem to require good diffusion properties as well as good nonlinear properties of the (uniform) masked S-box. These properties are not always easy to achieve, for example a uniform sharing of the AES S-box is not known and PRESENT is known to be weak against linear cryptanalysis. We find that there are non-trivial engineering challenges still ahead.

Chapter 6

Efficient Higher-Order Threshold implementations

In this chapter, we discuss techniques to improve the efficiency of higher-order threshold implementations while still allowing the security analysis of Chapter 5 to be applicable.

This chapter was made in collaboration with Tim Beyne, Adrián Ranea, Danilo Šijačić, Aein Rezaei Shahmirzadi, and Amir Moradi, and consists of two works published at Selected Areas in Cryptography, for which the preprint can be found online [11], and CHES, for which the preprint can also be found online [10], where I was a main author in both works.

6.1 Context

In Chapter 5, we introduced an analysis technique for the bounded-query probing model enabling us to create secure higher-order threshold implementations which require no fresh randomness for their computation. However, these threshold implementations are still quite inefficient. The example of the masked LED cipher required seven shares, whereas the minimum number of shares required for second-order security is three. It is also not always evident to verify the security of threshold implementations as this is currently done by hand using the wide-trail strategy. Finally, the theoretical model itself is quite unforgiving and unrealistic. It covers adversaries who can observe a large number of queries of the exact values on wires whereas in practice the attacker typically gets only

a few million noisy traces. The model also makes use of the glitch-extended probing model, through which the adversary sometimes gets several dozens of values per probe.

In this chapter, we introduce mechanisms to overcome these engineering challenges. We show that it is possible to reduce the number of shares and we put the theoretic model closer to practice by investigating a noisy version of the probing model and an improved glitch model. We also show how the changing of the guards method from Section 4.6.2 can be used in a higher-order security context and we show how one can amortise away even more randomness over multiple masked queries. We then apply these techniques again on LED and to create a second-order masking of the AES which requires almost no fresh randomness.

6.2 A Noisy Probing Model

In order to improve the efficiency of our maskings, we first require a more realistic adversary model. To that end, we relax the probing model to a noisy version. This makes a difference in the bounded-query security model from Section 2.6.3 as we adapt the model by changing the oracle. More specifically, we extend the notion of a probe to a *noisy probe*. Instead of giving back the exact values on the wire/or bundle, the noisy probe returns a *noisy leakage function* of the values. The advantage of a noisy t-threshold probing adversary $\mathcal{A}$ will be denoted by $\mathrm{Adv}_{\mathsf{noisy}\ t\text{-}\mathsf{thr}}(\mathcal{A})$.

The noisy probing model resembles the *noisy leakage model* first introduced by Chari *et al.* [30] and extended by Prouff and Rivain [86]. The main difference between the two models is in the information given to the adversary. In the noisy leakage model, the adversary is given a noisy function of all wire values in the circuit. In our model, as in the (glitch-extended) probing model, an adversary can only probe the circuit locally. However, unlike in the probing model, the adversaries' probes reveal only a noisy leakage function of the wire values. That makes the model similar to the noisy probing model from Dziembowski *et al.* [57]. However, the models differ in the way noisy leakage functions are defined. In addition, as opposed to the model of Dziembowski *et al.*, our model is purely information-theoretic, non-asymptotic, and limits the number of queries that can be made by the adversary. Moreover, we apply the noisy probing model to masking applications instead of re-keying applications.

6.2.1 Noisy Leakage Functions

In order to introduce the noisy probing model, it is necessary to introduce the notion of *noisy leakage functions*. Let d and $m_1, \ldots, m_d$ be positive integers. In Definition 6.1, the Hamming weight of a vector $x = (x_1, \ldots, x_d) \in \prod_{i=1}^{d} \mathbb{F}_2^{m_i}$ is denoted by $\mathsf{wt}(x) = |\{1 \leq i \leq d \mid x_i \neq 0\}|$. Furthermore, the set of vectors of weight i will be denoted by $B_d(i)$, *i.e.* the *Hamming circle* of radius i:

$$B_d(i) = \{x \in \textstyle\prod_{i=1}^{d} \mathbb{F}_2^{m_i} \mid \mathsf{wt}(x) = i\}\,.$$

Definition 6.1 provides a quantitative description of noisy leakage functions that will be useful to obtain our main theoretical result regarding the noisy probing model, *i.e.* Theorem 5.2. In this definition, $\mathbf{f}$ is a random function over the set of functions from $\prod_{i=1}^{d} \mathbb{F}_2^{m_i}$ to Ω. For example, in the Hamming weight leakage model with additive Gaussian noise $\Omega = \mathbb{R}$. The function $q_{\mathbf{f}}$ can be interpreted as a measure of similarity between the probability density functions $p_{\mathbf{f}(x_1)}$ and $p_{\mathbf{f}(x_2)}$ of the noisy leakage under secrets x_1 and x_2. The reciprocal noise parameters $\lambda_1, \ldots, \lambda_d$ then upper bound the Euclidean norm of the restriction of the Fourier transform of $q_{\mathbf{f}}$ to Hamming circles of successively increasing weights.

Definition 6.1 (Noisy leakage function)**.** Let $\Omega \subseteq \mathbb{R}^n$ be a measurable set. A dth order $(\lambda_1, \ldots, \lambda_d)$-noisy leakage function $\mathbf{f}$ is a random function from $\prod_{i=1}^{d} \mathbb{F}_2^{m_i}$ to Ω such that

$$1/\lambda_i \geq \frac{1}{|B_d(i)|} \left(\textstyle\sum_{u,v \in B_d(i)} \widehat{q}_{\mathbf{f}}(u,v)^2 \right)^{1/2},$$

where $\widehat{q}_{\mathbf{f}}$ is the Fourier transform of

$$q_{\mathbf{f}}(x_1, x_2) = \int_{\Omega} \frac{p_{\mathbf{f}(x_1)}(y)\, p_{\mathbf{f}(x_2)}(y)}{\mathbb{E}\,|\mathbf{f}^{-1}(y)|}\,\mathrm{d}y\,,$$

with $p_{\mathbf{f}(x)}$ the probability density function of $\mathbf{f}(x)$.

The noise parameters $\lambda_1, \ldots, \lambda_d$ characterise the level of the noise, with λ_i in particular representing the 'noise-level' when values from i probes are combined. The definition of the noise parameters may at first appear to be unusual. However, it will prove to be an apt measure for the purposes of Theorem 5.2.

The real-world value of the probing model relies in large part on the premise that higher-order side-channel attacks require a (geometrically) increasing number of traces to perform. Hence, it must be the case that combining information from different probes increases the noise. The following lemma shows that such

an amplification of noise indeed occurs, provided that the leakage functions of the probes are independent. The latter assumption is commonly referred to as the independent-leakage assumption and was first stated by Dziembowski and Pietrzak [58].

Lemma 6.1 (Noise amplification)**.** *Let* $\mathbf{f}_1, \ldots, \mathbf{f}_d$ *be mutually independent* λ*-noisy first-order leakage functions from* $\mathbb{F}_2^{m_i}$ *to* Ω*. The random function* $\mathbf{g}$ *defined by*

$$\mathbf{g}(x_1, \ldots, x_d) = (\mathbf{f}_1(x_1), \ldots, \mathbf{f}_d(x_d)) .$$

is a $(\lambda, \lambda^2, \ldots, \lambda^d)$*-noisy leakage function of order* d *from* $\prod_{i=1}^{d} \mathbb{F}_2^{m_i}$ *to* Ω^d*.*

A noisy threshold probing adversary $\mathcal{A}$ will be said to have independent λ-noisy probes if its probes jointly yield a leakage function $\mathbf{g}$ of the form described in Lemma 6.1. The functions $\mathbf{f}_1, \ldots, \mathbf{f}_d$ can then be interpreted as the leakage functions for the individual probes.

6.2.2 Bound on the Advantage

In order to use the noisy probing model in practice, it is necessary to be able to bound the advantage of adversaries in terms of some properties of the masking that can be computed or estimated. For the original bounded-query probing model from Section 2.6.3, Theorem 5.2 provides such a bound in terms of the linear-cryptanalytic properties of the masking. However, this theorem is not applicable to the new noisy probing model. Hence, we provide the following generalisation. The reader can find the proof in the preprint [10].

Similar to Theorem 5.2, Theorem 6.2 below assumes that any probed wire value can be labelled as 'good' or 'bad'. The values labelled 'good' jointly reveal nothing about the secret. The 'bad' values may reveal secret information, but the leakage can be bounded in terms of $\lambda_1, \ldots, \lambda_t$ and $\varepsilon_1, \ldots, \varepsilon_t$. The parameters $\lambda_1, \ldots, \lambda_t$ are determined by physical aspects such as the leakage model and noise level. The parameters $\varepsilon_1, \ldots, \varepsilon_t$ are instead determined by the mathematical properties of the masking.

Theorem 6.2. *Let* $\mathcal{A}$ *be a noisy* t*-threshold-probing adversary for a circuit* C*. Take* $\lambda_1, \ldots, \lambda_t \geq 1$*, and* $\varepsilon_1, \ldots, \varepsilon_d \leq 1$ *as non-negative real numbers. Assume that for every query made by* $\mathcal{A}$ *on the oracle* $\mathcal{O}^b$ *with result* $\mathbf{z}$*, there exists a partitioning (depending only on the probe positions) of the probed wire values into two random variables* $\mathbf{x}$ *('good') and* $\mathbf{y}$ *('bad') such that*

1. *The noisy leakage function* $\mathbf{f}$ *such that* $\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathbf{y})$ *is* $(\lambda_1, \ldots, \lambda_t)$*-noisy.*

2. *The conditional probability distribution* $p_{\mathbf{y}|\mathbf{x}}$ *satisfies* $\mathbb{E}_{\mathbf{x}}\|\widehat{p}_{\mathbf{y}|\mathbf{x}}\mathbb{1}_{B_t(d)}\|_2^2 \leq \varepsilon_d$ *for all* $d \in \{1, \ldots, t\}$*, with* $\mathbb{1}_{B_t(d)}$ *the indicator function of the set* $B_t(d)$ *and* $\widehat{p}_{\mathbf{y}|\mathbf{x}}\mathbb{1}_{B_t(d)}$ *the restriction of* $\widehat{p}_{\mathbf{y}|\mathbf{x}}$ *to this set.*

3. *Any* t*-threshold-probing adversary for the same circuit* C *and making the same oracle queries as* $\mathcal{A}$*, but which only receives the 'good' wire values (*i.e. *corresponding to* $\mathbf{x}$*) for each query, has advantage zero.*

The advantage of $\mathcal{A}$ *can be upper bounded as*

$$\mathrm{Adv}_{\mathsf{noisy}\ t\text{-}\mathsf{thr}}(\mathcal{A}) \leq \max_{1 \leq d \leq t} \sqrt{2q\,\varepsilon_d/\lambda_d}\,,$$

where q *is the number of queries to the oracle* $\mathcal{O}^b$*.*

6.2.3 Amortisation of Randomness

Later in Sections 6.4.2 and 6.5.6, we apply a technique to amortise randomness between several queries of a masked cipher. Meaning that we recover some randomness used in the state of one masked cipher to reuse in the next call. In this section, we investigate the security of a series of masked primitives which share randomness.

Suppose that the randomness is refreshed after every l calls to the masked cipher, *i.e.* after l blocks have been encrypted. To assess the security of such a construction, one can rely on a variant of the bounded query probing model from Section 2.6.3. Since the regular circuit oracle only generates new randomness every l calls, it is more convenient to consider an oracle for which each query corresponds to l invocations of the masked cipher. Equivalently, the adversary is given oracle access to a circuit that consists of l blocks which are only connected by the circuitry required to reuse randomness. The adversary is allowed to re-position the probes after each invocation of the masked circuit, so up to $2l$ probes per query are provided. However, only two probes per block are allowed. Moreover, we again extend the probes to be noisy. An adversary with this access structure may be called an l-block 2-noisy-threshold adversary. This is illustrated in Figure 6.1.

It is straightforward to adapt Theorem 6.2 to l-block 2-noisy-threshold adversaries. The only difference is in the admissible probe positions. However, when applying this result to determine the security bound, care must be taken in the labelling (as 'good' or 'bad') of probed values. In the $l = 1$ case, probed values resulting from two probes placed within the same S-box were marked as 'good'. This helps to avoid corner cases that do not threaten the probing security,

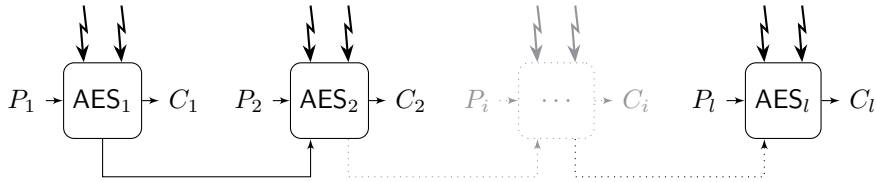


Figure 6.1: Illustration of the l-block 2-noisy-threshold probing model.

but that prevent the direct replacement of these values with uniform randomness in the proof of Theorem 6.2. For $l \geq 2$, the presence of probes in other blocks makes this labelling incorrect and an additional reduction is required before applying the theorem. In this initial step, the set of values obtained by placing both probes of a block in the same S-box is modified (expanded) to a set of values with a uniform random *marginal* distribution. By simply returning a noisy function of all but one of the input shares of the S-box, this is achieved without any loss in security. The modified values can then be labelled as 'bad'.

To complete the argument, an upper bound on $\|\widehat{p}_{\mathbf{z}} \mathbb{1}_{B_t(2)}\|_2^2$ must be determined. Here, $\mathbf{z}$ consists of the (modified) probed values conditioned on any values labelled as 'good'. This requires an analysis of linear trails, including trails that run across multiple blocks. Suppose that $\varepsilon_2 = 2^b \times c^2 \times \lambda_2$, where b is the maximum number of bits seen by any pair of noisy probes (possibly in different blocks), c^2 an upper bound on the squared correlation of any linear approximation between such a pair of probes, and λ_2 the noise on the pair of probes. Note that Lemma 6.1 is not necessarily applicable between the pairs of noisy probes since the noisy functions can be dependent between the blocks. Grouping masks by the number of active noisy probes, and applying the piling-up principle [76, 98], one obtains the upper bound

$$\|\widehat{p}_{\mathbf{z}} \mathbb{1}_{B_t(2)}\|_2^2 \leq \sum_{n=2}^{2l} \binom{2l}{n} \varepsilon_2^{n-1} = \varepsilon_2^{-1}\left[(1+\varepsilon_2)^{2l} - (1+2l\,\varepsilon_2)\right] \leq l(2l-1)\varepsilon_2\,.$$

It follows that a second-order noisy probing adversary making a total of q queries has its advantage bounded by

$$\mathrm{Adv}_{l\text{-}\mathsf{blk},\text{noisy 2-}\mathsf{thr}}(\mathcal{A}) \leq \sqrt{2l-1} \times \sqrt{2\,q\,\varepsilon_2}\,.$$

That is, a factor of $\sqrt{2l-1}$ is lost when randomness is reused over l masked cipher calls.

6.3 Masking Techniques

In this section we go over two masking techniques to improve the efficiency of solutions compared to the strategy proposed in Chapter 5. The first technique splits each quadratic function into two stages and uses non-completeness and uniformity properties. The second method builds further on the first by pairing masked S-boxes to reduce their latency.

6.3.1 Technique 1: Non-Completeness over Two Stages

In order to apply our theoretical analysis to reduce randomness, the masking of the S-box must have a sufficiently small maximum absolute correlation in the sense of Definition 5.3. To that end, we use the first-order non-complete and uniform maskings by Bilgin *et al.* [17] as a starting point to create low-randomness second-order maskings.

In Chapter 5, we required that each stage in the masked design is second-order non-complete and uniform. This requirement comes at a high price in terms of the number of shares and thus in area cost. Instead, we relax it by making sharings which are uniform only every two stages. Since we will reuse randomness between S-boxes, we additionally require that our sharings still achieve first-order non-completeness after two stages.

We start from the non-complete and uniform sharings of Bilgin *et al.* and divide them into two stages such that each stage is second-order non-complete using the ring re-masking technique of Reparaz *et al.* [88], but we only refresh the cross terms. We illustrate our approach by giving a uniform masking of an AND-XOR gate, which maps (a, b, c) to $ab + c$. The masking uses random bits $r_0, \ldots, r_5$.

$$
\begin{array}{lll}
f_0(a_0, b_0, c_0) = a_0b_0 + c_0 & \to x'_0 & \\
f_1(a_0, b_1) \quad = a_0b_1 + r_5 + r_0 & \to x'_1 & \to x'_0 + x'_1 + x'_2 = x_0 \\
f_2(a_1, b_0) \quad = a_1b_0 + r_0 + r_1 & \to x'_2 & \\
\hline
f_3(a_1, b_1, c_1) = a_1b_1 + c_1 & \to x'_3 & \\
f_4(a_1, b_2) \quad = a_1b_2 + r_1 + r_2 & \to x'_4 & \to x'_3 + x'_4 + x'_5 = x_1 \\
f_5(a_2, b_1) \quad = a_2b_1 + r_2 + r_3 & \to x'_5 & \\
\hline
f_6(a_2, b_2, c_2) = a_2b_2 + c_2 & \to x'_6 & \\
f_7(a_0, b_2) \quad = a_0b_2 + r_3 + r_4 & \to x'_7 & \to x'_6 + x'_7 + x'_8 = x_2 \\
f_8(a_2, b_0) \quad = a_2b_0 + r_4 + r_5 & \to x'_8 &
\end{array}
$$

Each arrow denotes a register stage. Note that the first output share equals $a_0b_0 + a_0b_1 + a_1b_0 + c_0 + r_1 + r_5$ which is still non-complete, but now also contains nonlinear terms.

In order to ensure the second-order non-completeness of the second stage of the sharing, we used fresh randomness in a ring refreshing configuration. Using the theory from Section 6.2 we can reuse this randomness in every S-box. This can be seen as follows. Probing two S-boxes with jointly uniformly shared inputs as in Figure 6.2 only gives a non-complete set of input shares due to the masking's properties (even without randomness).

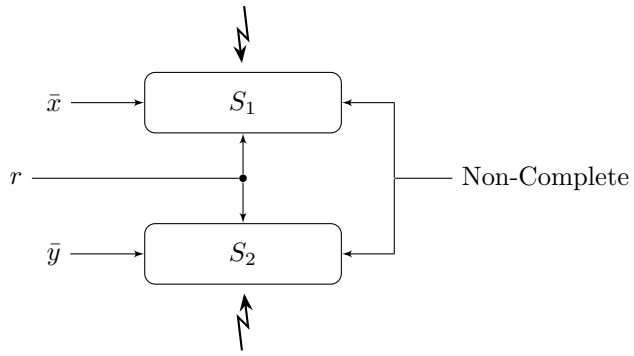


Figure 6.2: Two masked S-boxes sharing the same randomness. Probes are indicated by lightning signs. Since both S-boxes are non-complete without the randomness r, the probes do not return secret information.

6.3.2 Technique 2: Paired Masked S-boxes

The technique from Section 6.3.1 allows us to reduce the number of shares and still create randomness-free second-order sharings. However, each quadratic function requires two register stages and this increases the latency of the masking.

Removing the second register layer breaks the non-completeness property over two stages that was outlined above, causing a failure to achieve second-order probing security. Adding randomness to the design is not an option as we intend to reuse all randomness in each S-box. Instead, we pair two S-boxes that use each other's inputs to help achieve second-order probing security. We also use different randomness for each S-box in the pair, but this randomness can be re-used for each pair. The overall configuration is depicted in Figure 6.3.

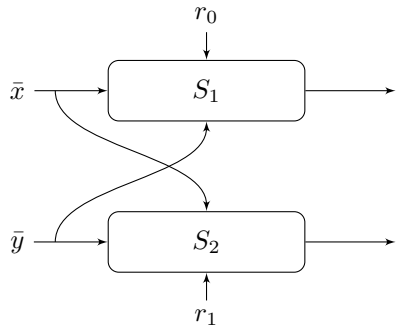


Figure 6.3: Two paired masked S-boxes.

Essentially, the idea is that if two probes are placed in the same pair of masked S-boxes, the randomness (r_0, r_1) ensures that the probes do not observe secret information. When probing two different pairs using the same randomness, the inputs from the paired S-boxes act as fresh randomness.

The above trick is similar to the changing of the guards technique from Section 4.6.2, as we are using the inputs of one masked S-box in another one. However, here we are not solving the uniformity of the sharing. Instead, we are using the inputs to ensure the non-completeness property of the sharing.

However, we should be careful with the above trick as the second property required to apply our cryptanalytic analysis is that the masking needs to have good diffusion. In case the output of the masked S-box depends on its paired second S-box, the diffusion properties of the masked cipher are altered. Instead, we add the inputs of a second S-box such that the dependency disappears after two stages. Below, we provide an example of the masking technique for two

paired AND-XOR gates:

$f_0(a_0, b_0, c_0) = a_0b_0 + c_0 + k_0 + l_0$	$\to x'_0$	
$f_1(a_0, b_1) = a_0b_1 + r_5 + r_0 + k_0$	$\to x'_1$	$x'_0 + x'_1 + x'_2 = x_0$
$f_2(a_1, b_0) = a_1b_0 + r_0 + r_1 + l_0$	$\to x'_2$	
$f_3(a_1, b_1, c_1) = a_1b_1 + c_1 + k_0 + l_0$	$\to x'_3$	
$f_4(a_1, b_2) = a_1b_2 + r_1 + r_2 + k_0$	$\to x'_4$	$x'_3 + x'_4 + x'_5 = x_1$
$f_5(a_2, b_1) = a_2b_1 + r_2 + r_3 + l_0$	$\to x'_5$	
$f_6(a_2, b_2, c_2) = a_2b_2 + c_2 + k_0 + l_0$	$\to x'_6$	
$f_7(a_0, b_2) = a_0b_2 + r_3 + r_4 + k_0$	$\to x'_7$	$x'_6 + x'_7 + x'_8 = x_2$
$f_8(a_2, b_0) = a_2b_0 + r_4 + r_5 + l_0$	$\to x'_8$	

$f_0(k_0, l_0, m_0) = k_0l_0 + m_0 + a_0 + b_0$	$\to y'_0$	
$f_1(k_0, l_1) = k_0l_1 + r_{11} + r_6 + a_0$	$\to y'_1$	$y'_0 + y'_1 + y'_2 = y_0$
$f_2(k_1, l_0) = k_1l_0 + r_6 + r_7 + b_0$	$\to y'_2$	
$f_3(k_1, l_1, m_1) = k_1l_1 + m_1 + a_0 + b_0$	$\to y'_3$	
$f_4(k_1, l_2) = k_1l_2 + r_7 + r_8 + a_0$	$\to y'_4$	$y'_3 + y'_4 + y'_5 = y_1$
$f_5(k_2, l_1) = k_2l_1 + r_8 + r_9 + b_0$	$\to y'_5$	
$f_6(k_2, l_2, m_2) = k_2l_2 + m_2 + a_0 + b_0$	$\to y'_6$	
$f_7(k_0, l_2) = k_0l_2 + r_9 + r_{10} + a_0$	$\to y'_7$	$y'_6 + y'_7 + y'_8 = y_2$
$f_8(k_2, l_0) = k_2l_0 + r_{10} + r_{11} + b_0$	$\to y'_8$	

In red, we denote the added randomness which can be re-used for each pair of AND-XOR sharings. In blue, we denote the paired gate's input. The diffusion of the masking remains unaffected, for example, the output $x_0 = a_0b_0 + a_0b_1 + a_1b_0 + c_0 + r_1 + r_5$ does not depend on (k_0, l_0).

6.4 Application to LED

In this section, we apply the noisy security model from Section 6.2 and the masking techniques from Section 6.3.1 and 6.3.2 to create a more efficient sharing of the LED compared to Section 5.6. We also create a sharing of the LED that generates as much randomness as it uses.

6.4.1 A Low-Latency Variant

In this section we apply the masking technique from Section 6.3 to create a three-shared masking of the LED which requires no fresh randomness. The masking is adapted such that it requires only a few register stages.

Masking The S-box S belongs to the cubic class $\mathcal{C}^4_{266}$. There are several ways to decompose the S-box into quadratic bijections, we chose the decomposition $S = A_3 \circ \mathcal{Q}^4_{12} \circ A_2 \circ \mathcal{Q}^4_{12} \circ A_1$ with A_1, A_2 and A_3 affine and $\mathcal{Q}^4_{12}$ quadratic. The affine maps are defined by

$$A_1 : (a,\ b,\ c,\ d) \mapsto (a,\ d,\ b,\ b + c + 1),$$

$$A_2 : (a,\ b,\ c,\ d) \mapsto (c + d,\ c,\ a,\ a + b),$$

$$A_3 : (a,\ b,\ c,\ d) \mapsto (a + d + 1,\ b + c,\ a + c + d,\ b + c + d + 1)\,.$$

In our first design, we make a second-order probing-secure masked version of the S-box in such a way that it remains first-order secure without fresh masks. This enables us to reuse the fresh masks in all S-boxes and in all rounds, which will be discussed in more detail in the security analysis paragraph below. To guarantee the security of the design, the outputs of A_1, A_2, and the second application of $\mathcal{Q}^4_{12}$ should be stored in registers. Otherwise, the first-order non-completeness would be violated. Additionally, the masking of $\mathcal{Q}^4_{12}$ also needs a register layer before compression, leading to a 5-stage design of the masked LED S-box.

To improve the latency, we can apply our second masking technique (introduced in Section 6.3.2) and remove two register layers to make a 3-stage design. To this end, we integrate the middle and output affine functions, A_2 and A_3, at the output of the quadratic functions. Namely, $S = G \circ F \circ A_1$ with $F = A_2 \circ \mathcal{Q}^4_{12}$ where

$$F : (a,\ b,\ c,\ d) \mapsto (bd + c + d,\ bd + c,\ a,\ bd + cd + a + b),$$

and $G = A_3 \circ \mathcal{Q}^4_{12}$ where

$$G : (a,\ b,\ c,\ d) \mapsto (a + d + 1,\ cd + b + c,\ bd + a + c + d,\ cd + b + c + d + 1).$$

Note that the composition of A_1 and $\mathcal{Q}^4_{12}$ contains a coordinate function containing all quadratic monomials in three input variables. Hence, it does not have a 3-share non-complete and uniform sharing. For this reason, we implement the input affine map A_1 separately such that the output of A_1 is stored in a register to ensure non-completeness. We then pair two S-boxes, following the second technique described in Section 6.3.2 and add randomness to the paired design. These random masks can be reused for each pair of S-boxes in the entire encryption.

The details of the sharings and their performance can be found in the preprint [10]. Other ciphers like PRINCE, SKINNY, and Midori are also implemented using the same techniques. In the performance comparison it can be seen that the performance of the sharings without fresh-randomness can

compete with the state-of-the-art maskings. This shows the practical viability of the theory from Chapter 5.

Security Analysis The security of one round of the construction was verified using the SILVER verification tool [71]. Additionally, since the key schedule is an affine function, a probe on the key-addition operation depends on at most one share of the key. As a result, we can consider all probe positions in one round and the shares of the key as 'good' values when applying Theorem 6.2. The same reasoning applies to the additional randomness that is reused across S-boxes.

Before Theorem 6.2 can be applied to obtain a bound on the security against noisy 2-threshold probing adversaries, the analysis of the 'bad' wire values must be completed. This is the most difficult part of the security analysis and requires determining ε_1 and ε_2. Due to the first-order security of the masking, $\varepsilon_1 = 0$. The value of ε_2 is determined by the correlation of linear approximations resulting from probes placed in two different rounds.

To upper bound the correlation of linear trails, the following upper bound on the maximum absolute correlation of linear approximations over the masked LED S-box will be used.

Lemma 6.3. *Let $\bar{S} : \mathbb{V}_a \to \mathbb{V}_b$ be any restriction of the sharing of S defined above. Denote its absolute correlation matrix by $|C^{\bar{S}}|$. For any $u, v \in \mathbb{F}_2^\ell / \mathbb{V}^\perp$ such that $u \neq 0$, it holds that $\left|C^{\bar{S}}_{u,v}\right| \leq 2^{-2}$.*

A single probe on the LED state results in a value that depends either on one column of the state (when placed at the `MixColumns` step) or on the input or output of two (paired) S-boxes in the same column. Hence, it suffices to consider masks (in the sense of linear cryptanalysis) that activate at most one column of the input and output state of the trail. As a result, we search for the activity pattern between two columns of the LED state activating the least number of S-boxes. As a result, this analysis is entirely similar to the one performed in Section 5.6.4 and 5.6.5. There, it was shown that all such two- and three-round trails have correlation zero. For more than three rounds, the best linear trails span four rounds and activate 24 S-boxes.

Thus, for probes placed in rounds i and $i+r$ with $r \geq 3$, the relevant linear trails all have at least 24 active S-boxes. Hence, using Lemma 6.3, the correlations of these trails are bounded by 2^{-48}. By the piling-up principle, we expect a similar bound for the correlation of linear approximations. It then follows that the 2-norm of the nontrivial Fourier coefficients of the observed bits $\mathbf{z}$ can be

upper bounded by

$$\varepsilon_2 := \|\widehat{p}_{\mathbf{z}} \mathbb{1}_{B_2(2)}\|_2^2 \leq |\operatorname{supp} \widehat{p}_{\mathbf{z}}| \, \|\widehat{p}_{\mathbf{z}} \mathbb{1}_{B_2(2)}\|_\infty^2 \leq 2^{64} \, 2^{-96} = 2^{-32},$$

where we have used the inequality $|\operatorname{supp} \widehat{p}_{\mathbf{z}}| \leq 2^{24}$, which follows from the fact that the observed value $\mathbf{z}$ consists of at most 24 bits in the glitch-extended probing model: if a coordinate in $\bar{S}$ is read, at most 12 shares are learned; if an output of the shared linear layer is probed, at most 32 shares are observed.

The above analysis motivates the following security claim, which relies only on the accuracy of the piling-up principle (that resulted in the estimate $\varepsilon_2 \leq 2^{-32}$).

Security Claim 2. Let $\mathcal{A}$ be a noisy 2-threshold probing adversary for the masking of LED described in this section. If $\mathcal{A}$ makes at most q queries and the probes of $\mathcal{A}$ are independent and λ-noisy, then the advantage of $\mathcal{A}$ is bounded by (assuming piling-up)

$$\mathrm{Adv}_{\mathsf{noisy\ 2\text{-}thr}}(\mathcal{A}) \leq \sqrt{\frac{q}{\lambda^2 \, 2^{31}}} \, .$$

Due to the above security bound, the advantage of any noisy-probing adversary making at most 2^{27} queries is at most 2^{-8} as long as $\lambda \geq 2^6$.

6.4.2 A Randomness-Free Variant

We create a second version of the three-shared LED above. Namely a version which generates as much randomness as it requires using an amortisation technique and the theory from Section 6.2.3.

Masking We use the three-shared LED masking from Section 6.4.1 (either the design using the technique from Section 6.3.1 or the technique from Section 6.3.2). However, we add an extra register stage just before the diffusion layer of LED. This increases the latency of the masking but it improves its security. Looking back at the security analysis above, the analysis stays the same apart from the number of bits an adversary can view using one probe. Namely, at most 12 bits can be viewed as opposed to the 32 bits in the previous design. This provides a 40 bit improvement in the bound.

Security Claim 3. Let $\mathcal{A}$ be a noisy 2-threshold probing adversary for the masking of LED described in this section. If $\mathcal{A}$ makes at most q queries and the probes of $\mathcal{A}$ are independent and λ-noisy, then the advantage of $\mathcal{A}$ is bounded by (assuming piling-up)

$$\mathrm{Adv}_{\mathsf{noisy\ 2\text{-}thr}}(\mathcal{A}) \leq \sqrt{\frac{q}{\lambda^2 \, 2^{71}}} \, .$$

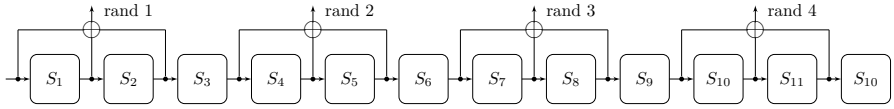


Figure 6.4: Extraction of randomness from a masked LED to create the random bits "rand 1" to "rand 4". The ith step function (comprising four rounds) of the masked LED is denoted by S_i.

We then further adapt the masking by amortising randomness to be used for the next encryption call. The technique works by extracting the randomness from the masked state in certain rounds. Here extraction means taking two out of three shares of each bit of the state for subsequent use. Directly using the extracting state bits as randomness in the next masked cipher call would be insecure in the probing model. Instead, we extract randomness from multiple rounds and add it together. In total the shared state size is extracted four times to generate either 408 or 456 random bits, namely the number of random bits the design requires. This process is depicted in Figure 6.4.

As a result, the sharing does not need an additional random number generator (deterministic or non-deterministic) to function, it only needs an initial seed of randomness. This saves both in area costs and improves the security since the analysis (detailed further-on) covers the entire primitive as opposed to analyses which exclude the random number generator. Thus, we cover the threats outlined in Section 3.4 on the integration of a random number generator.

The generation of randomness via the extraction method outlined above is quite cheap (in energy cost) as it requires only two XORs and a register to generate one random bit nearing the cost of using an LFSR as outlined in Section 3.5. However, the area cost of the extraction is high as it is essentially rolled out. Approximately this method requires an additional 3500 GE. It would be interesting to further develop this method to reduce these costs.

Security Analysis The largest part of the security analysis was already performed with the previous sharing in Section 6.4.1. We investigate trails over multiple blocks of the masked LED where trails go over the randomness extraction mechanism. Due to the property of the diffusion layer activating at least 24 S-boxes over four rounds of LED, when activating a cell in "rand i", even when placing two additional probes, at least 24 masked S-boxes are activated. As a result, even over multiple blocks $\varepsilon_2 \leq 2^{72}$. From the security analysis of Section 6.2.3, we find the following bound.

Security Claim 4. For the masked LED with randomness reuse, the following

bound on the advantage of a noisy 2-threshold probing adversary over l blocks $\mathcal{A}$ (assuming piling-up) in the noisy probing model is claimed:

$$\mathrm{Adv}_{l\text{-blk},\mathsf{noisy\ 2\text{-}thr}}(\mathcal{A}) \leq \sqrt{2l-1} \times \sqrt{\frac{q}{\lambda^2\, 2^{71}}}\,.$$

Assuming the adversary can only take up to 2^{27} queries, we set $l = 2^{27}$. Taking $q = 2^{27}$, an upper bound of the advantage of the adversary is 2^{-8} without noise. Including the noise would only improve this bound. This means we can safely keep reusing the randomness between blocks making the masking of the LED entirely randomness-free.

The above security claim should be investigated further as the piling-up principle might not be applicable since the randomness extraction makes the entire masked cipher non-uniform. Due to this, there may be a large number of trails with a similar absolute correlation for a given mask on the extracted randomness.

6.5 Application to AES

In this section, we apply the theory from Chapter 5 to a second-order masked AES. We have already seen that our analysis can create efficient sharings of lightweight primitives such as LED, PRINCE, SKINNY, and Midori. In this section we create a second-order sharing of the AES which requires low-randomness.

6.5.1 Masking

We first provide the sharing details of the second-order masked AES. Later, we go in detail how the S-box sharing can be optimised and how to changing of the guards method is used.

Masking State and Key For the sharing of the AES state and key, we use classical Boolean masking. The 128-bit state is shared using four shares per bit, requiring a total of $128 \times 3 = 384$ random bits. The 128-bit key is also shared using four shares, and this also costs 384 random bits. Finally, we extend the state by an additional column containing three shares of randomness. This requires an additional $32 \times 3 = 96$ random bits and is necessary for the *guards in formation* technique that will be described in Section 6.5.3. It will be used over the rows of the state for the S-box layer.

An overview of the shared AES round function is shown in Figure 6.5. The following sections discuss further aspects of this sharing.

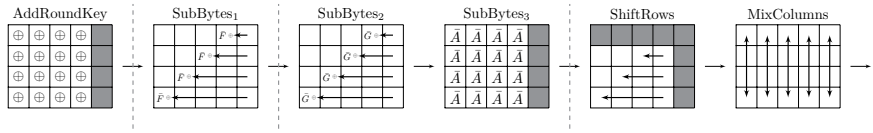


Figure 6.5: One round of the masked AES. The locations of the registers are indicated by dashed lines. The nonlinear operations require an additional register stage, which is not shown on the figure. Grey cells remain unchanged through the operation.

Sharing the S-Box The AES S-box consists of an inversion S over $\mathbb{F}_{2^8}$ and an affine layer A. Similar to the linear layers of the AES, the affine layer is masked using a share-wise approach where the constant is added to the first share. Following the work by Wegener *et al.* [101], the inversion ($x \mapsto x^{254}$) is decomposed into two cubic functions. Specifically, we consider

$$x^{254} = \left(x^{26}\right)^{49} = G(F(x)) \,,$$

where $F : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8} : x \mapsto x^{26}$ and $G : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8} : x \mapsto x^{49}$.

We use the masking technique of Section 6.3.1 to share the S-box. Non-complete sharings of F and G can be achieved by using four input shares. To achieve second-order probing security, the sharings $\bar{F}$ and $\bar{G}$ are split into two stages separated by a register. This is depicted in Figure 6.6. Specifically, $\bar{F} = \bar{F}_2 \circ \bar{F}_1$ and $\bar{G} = \bar{G}_2 \circ \bar{G}_1$ with $\bar{F}_1$, $\bar{F}_2$, $\bar{G}_1$ and $\bar{G}_2$ second-order non-complete. In particular, $\bar{F}_2$ and $\bar{G}_2$ merely implement a linear compression of shares into four output shares. To ensure the second-order probing security of $\bar{F}$ and $\bar{G}$, randomness is added at the end of the stages $\bar{F}_1$ and $\bar{G}_1$.

Finally, the sharing is made uniform using the changing of the guards approach of Daemen [35]. In Section 4.6.2, it was shown how the additional randomness that is used for this can be reused without sacrificing uniformity.

Key Schedule The key schedule is masked similar to the state. Meaning that linear layers are masked share-wise and the S-box is masked using the method above. We instantiate the additional cell due to the changing of the guards technique with fresh randomness in every evaluation. This costs 24 random bits for every $\bar{F}$ or $\bar{G}$ layer for a total randomness cost of $20 \times 24 = 480$ bits plus an additional 384 random bits for the initial sharing of the master key. The 456

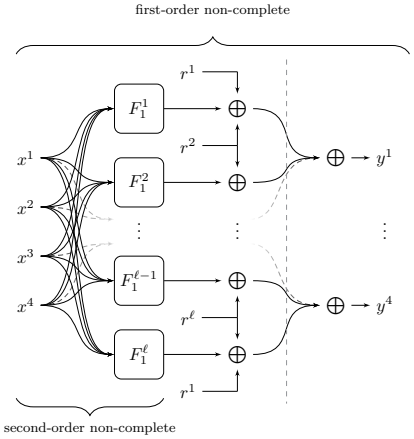


Figure 6.6: Sharing of the cubic function F (or G). The Boolean functions $F_1^1, \ldots, F_1^\ell$ are the shares of the first layer $\bar{F}_1$ (or $\bar{G}_1$). The grey dashed line denotes a register stage.

random bits to ensure the local second-order probing security of the masked S-box can be re-used from the masked S-boxes in the state as will be explained in Section 6.5.4.

6.5.2 Optimising the S-Box Sharing

As explained above, we use the masking technique outlined in Section 6.3.1 to mask the S-box. This section shows how to minimise the randomness costs by reducing the number of output shares in the second-order non-complete expansion layers $\bar{F}_1$ and $\bar{G}_1$.

We start with a straightforward method to choose $\bar{F}_1$ and $\bar{G}_1$. Denote the output bits of $\bar{F}$ (similarly $\bar{G}$) by $y_i^1, y_i^2, y_i^3, y_i^4$ for $i \in \{1, ..., 8\}$. Each output bit is a function of at most three shares of each input bit due to $\bar{F}$ being non-complete. One can again find a second-order non-complete sharing for each of these three-shared functions. This would result in a decomposition of $\bar{F}$ that is second-order non-complete in each stage, while maintaining the first-order non-completeness of $\bar{F}$.

The above method has the downside that it results in functions $\bar{F}_1$ with a large number of shares. This can be optimised further. Instead of using a second-order non-complete sharing of each F^i (or G^i), we can re-share some functions using a first-order non-complete covering scheme. The optimised

covering scheme is shown in Table 6.1. The third column shows which input shares can be combined in $\bar{F}_1$ (or $\bar{G}_1$). For example, F^1 is re-shared such that each output bit can use only either the first and second share of an input bit or the first and third. This covering scheme can be verified to be second-order non-complete. For example, it is clear that one probe can never learn the second and third share of an input bit.

Table 6.1: This table depicts the share dependencies of an input bit for $\bar{F}$, $\bar{G}$ or $\bar{F}_1$, $\bar{G}_1$. The table also shows which random bits are added to the output of $\bar{F}_1$ or $\bar{G}_1$.

Output share	$\bar{F}$ or $\bar{G}$	$\bar{F}_1$ or $\bar{G}_1$	Used random bits
1	$\{x^1, x^2, x^3\}$	$\{x^1, x^2\}, \{x^1, x^3\}$	$r_1, ..., r_{96}$
2	$\{x^2, x^3, x^4\}$	$\{x^2\}, \{x^3\}, \{x^4\}$	$r_{97}, ..., r_{456}$
3	$\{x^1, x^3, x^4\}$	$\{x^1, x^3\}, \{x^1, x^4\}$	$r_1 + r_{97}, ..., r_{96} + r_{192}$
4	$\{x^1, x^2, x^4\}$	$\{x^1, x^2\}, \{x^1, x^4\}$	$r_1 + r_{193}, ..., r_{96} + r_{288}$

While Table 6.1 shows a second-order non-complete covering scheme for $\bar{F}_1$ and $\bar{G}_1$, we still need to share $\bar{F}$ or $\bar{G}$ following those requirements. For this, we use the covering schemes from the work by Bozilov [20]. That work provides a covering such that each function F^1, F^3, and F^4 is re-shared to have 12 output shares and F^2 can be re-shared to have 45 output shares. Thus, we have a total of $45 \cdot 8 + 24 \cdot 12 = 648$ output shares for $\bar{F}_1$ (similarly $\bar{G}_1$).

In order to ensure the second-order probing security of the sharings of $\bar{F}$ and $\bar{G}$, a total of 648 random bits are added to the second-order non-complete sharing before re-compression. We can reduce this number by observing that the security condition boils down to requiring that all values seen by the probing adversary need to be masked with a unique random bit. In Table 6.1, we show a reduction of this randomness to 456 bits. By probing, an adversary can see all random bits related to an output share listed in the table. One can see that, even if two output shares are probed, all observed bits in the expansion are masked with a unique random bit. For example, probing output shares 2 and 3, the adversary observes $r_{97}, ..., r_{456}$ and $r_1 + r_{97}, ..., r_{96} + r_{193}$. The observations are then still masked by the unique random bits $r_{97}, ..., r_{456}$ and $r_1, ..., r_{96}$.

6.5.3 Guards in Formation

This section discusses the application of the changing of the guards technique to the S-box layer. Recall from Section 4.6.2 that three out of four shares of one

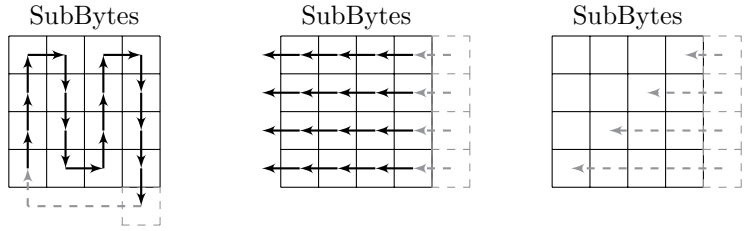


Figure 6.7: Three examples of diffusion patterns using the changing of the guards technique. Additional cells are outlined by dashed grey lines. The third example chains all cells in a row.

input (b^1, b^2, b^3) are used to re-mask the other branch (a'^1, a'^2, a'^3, a'^4) in the Feistel structure. However, when used in a straightforward way, this operation is not second-order non-complete as three shares are used to mask the fourth share of the other branch. In order to make the re-masking operation second-order probing secure, it can be spread across the two stages of the sharing $\bar{F}$ (or $\bar{G}$).

It is also important to consider how the changing of the guards structure links the different cells of the state. As the cryptanalytic properties of our masking affect its security, the diffusion resulting from the linear layer plays an important role. However, from the perspective of linear-cryptanalysis, placing a changing of the guards structure over the S-box layer reduces diffusion. To improve diffusion while keeping the cost minimal, we look for inspiration in the Rijndael-160 cipher [38].

A traditional application of the changing of the guards method would result in one additional state cell. This extra cell is instantiated with a random sharing of zero at the start of execution. One has several options on how to lay out the changing of the guards structure in this case. One example is shown in the leftmost illustration in Figure 6.7. Since the changing of the guards method now mixes shares from different cells, it affects the diffusion of the shared cipher. Using the security analysis tool that will be introduced in Section 6.5.4, one finds that the example in Figure 6.7 results in linear trails with few active S-boxes. Consequently, the security bound obtained using Theorem 6.2 would not be satisfactory.

Instead, we shorten the chain of cells linked by the changing of the guards structure by increasing the number of additional cells. Specifically, the changing of the guards method is applied to the S-boxes in each of the rows of the state independently. This is illustrated in the middle illustration of Figure 6.7. Furthermore, the diffusion properties over the enlarged state are improved by also applying a `MixColumns` operation over the four extra cells. Finally, to

complete the analogy with Rijndael-160, a 'shift' is introduced in the changing of the guards structure. The shifting offset depends on the row number as depicted in the rightmost part of Figure 6.7. The final result is shown in Figure 6.5.

6.5.4 Security Analysis

This section determines an upper bound on the advantage of second-order probing adversaries for the masked AES construction. The security analysis is split into two parts, corresponding to the analysis of 'good' and 'bad' labelled values in Theorem 6.2. In the first part, we consider probes placed within a single round of the masked cipher. In the second part, wire values resulting from probes placed in different rounds are analysed. The first part of the analysis uses standard probing security arguments whereas the second relies on linear cryptanalysis.

Single Round In this section, we argue that one round of the masked AES is second-order probing secure. Recall that, without loss of generality, it may be assumed that all probes are placed right before a register stage. Probes placed in linear layers only return one share per input bit. Probing two linear layers is trivially secure, and probing a linear layer and an S-box can be reduced to the following cases where two S-boxes are probed. Hence, only two cases must be considered: both probes placed within the same shared S-box (either $\bar{F}$ or $\bar{G}$), or probes positioned in two different shared S-boxes.

For the first case, consider that the adversary places both probes in $\bar{F}$ (the argument is the same for $\bar{G}$). If both probes are placed in $\bar{F}_1$, the adversary does not receive all shares of any secret bit since $\bar{F}_1$ is a second-order non-complete sharing. Thus, consider the case where the adversary probes the compression layer $\bar{F}_2$. As shown in Table 6.1, each bit at the input of the expansion layer is masked with a unique random bit. Thus, whatever the choice of the second probe position, the adversary cannot infer any information about an input secret of $\bar{F}$. As a result, the sharings $\bar{F}$ and $\bar{G}$ are second-order probing secure.

Consider the second case, *i.e.* the two probes are placed in different S-boxes. If both probes are placed in the $\bar{F}$-part of the S-box sharings (similarly $\bar{G}$) then, due to first-order non-completeness, the adversary does not learn any secret bits. Hence, we consider the case where one probe is placed in $\bar{F}$ and one is placed in $\bar{G}$. This case is illustrated in Figure 6.8. In this figure, $\bar{b}$ denotes the additional branch used for the changing of the guards technique. This branch $\bar{b}$ is a uniform random sharing of zero and is used to re-mask the output of $\bar{F}$. From the probe in $\bar{F}$, the adversary learns at most one output share of $\bar{F}$. We

argue that the probe in $\bar{G}$ cannot reveal all of the other output shares. The argument is based on the consistent use of the same covering scheme for both sharings:

- Due to the second-order non-completeness of the expansion layer $\bar{G}_1$ of $\bar{G}$, probing the layer $\bar{G}_1$ does not reveal any of the other three output shares of $\bar{F}$.
- When probing the compression layer $\bar{G}_2$ of $\bar{G}$, the resulting values are masked by the randomness $\bar{r}$. The adversary can only see the same bits of $\bar{r}$ as obtained from the probe in $\bar{F}$ when the indices of the probed output shares of $\bar{F}$ and $\bar{G}$ are the same. The covering scheme used for non-completeness (second column of Table 6.1) guarantees that the adversary does not obtain all of the output shares of $\bar{F}$.

Hence, this case is also second-order probing secure.

The above security argument also holds for the key schedule, as it uses the same masked S-box. In particular, the four S-boxes in one round of the key schedule can be considered to be parallel to the 16 S-boxes in the round function.

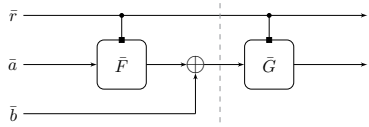


Figure 6.8: Depiction of the consecutive $\bar{F}$ and $\bar{G}$ functions.

Multiple Rounds Following Theorems 6.2 and 5.4, we argue the second-order noisy probing security of multiple rounds of the masked AES by bounding the correlation of all linear trails resulting from two probes placed in different rounds. Specifically, by Theorem 5.4, bounding the correlation of linear approximations in the masked cipher results in an upper bound on $\|\widehat{p}_{\mathbf{z}}\mathbb{1}_{B_2(2)}\|_2^2$ where $p_{\mathbf{z}}$ is the probability distribution of the probed values. In turn, by Theorem 6.2, this provides a bound on the advantage of second-order noisy probing adversaries.

All of the randomness which is re-used across S-boxes, can be labelled as 'good' in the terminology of Theorem 6.2. This is safe, since an adversary only probing these values cannot obtain any secret information. Consequently, in the linear cryptanalysis of the masked cipher, these random bits must be considered to be constant. Since F and G are shared in a non-complete way even without the use of randomness, we can assume that any probe placed by the adversary

results in a set of non-complete input shares. As a result, $\varepsilon_1 = 0$ and only linear approximations with nonzero input and output masks must be considered.

To ensure the security of the masking, all linear approximations over the sharings $\bar{G}$ and $\bar{F}$ should have a low absolute correlation. Verifying this property is slow since the masked S-box has 32 input bits. Using optimised verification software, it takes 1500 core hours on an Intel(R) Xeon(R) Gold 6230 CPU with a clock frequency of 2.10 GHz to compute the linear approximation table of a sharing for one input secret (one restriction). We use the direct four-sharing of F and G as defined in the thesis of Bilgin [14, pg. 36]. The search revealed sharings of the F and G with the following properties.

Claim 1. Let $\bar{F} : \mathbb{V}_a \to \mathbb{V}_b$ be any restriction of the sharing of F. Denote its absolute correlation matrix by $|C^{\bar{F}}|$. For any $u, v \in \mathbb{F}_2^\ell / \mathbb{V}^\perp$ not both equal to zero, it holds that $\left|C^{\bar{F}}_{u,v}\right| \leq 2^{-3}$ and, moreover, $\left|C^{\bar{F}}_{0,v}\right| \leq 2^{-3.8}$.

Claim 2. Let $\bar{G} : \mathbb{V}_a \to \mathbb{V}_b$ be any restriction of the sharing of G. Denote its absolute correlation matrix by $|C^{\bar{G}}|$. For any $u, v \in \mathbb{F}_2^\ell / \mathbb{V}^\perp$ not both equal to zero, it holds that $\left|C^{\bar{G}}_{u,v}\right| \leq 2^{-2.6}$ and, moreover, $\left|C^{\bar{G}}_{0,v}\right| \leq 2^{-4}$.

To upper bound the maximum absolute correlation of linear trails between the observed values (corresponding to nonzero masks), we use a slight refinement of the standard wide-trail type argument. That is, we search for the best trail activity patterns over the masked AES, but we take into account both cases in Claims 1 and 2.

As opposed to the security analysis in Section 5.6, here we cannot go through all activity patterns by hand. Instead, we make use of automated tools. Specifically, we encode this search problem as a sequence of Satisfiability Modulo Theories (SMT) problems in the bit-vector theory. These problems are then solved with the off-the-shelf SMT solver Boolector [83]. A similar approach was originally used by Mouha *et al.* [80] to search for activity patterns of the unmasked AES using Mixed Integer Linear Programming. Whereas the approach of Mouha *et al.* is to search for activity patterns with a minimal number of active S-boxes, our approach is to look for activity patterns with maximal absolute correlation.

To model the correlation of the shared S-boxes, we consider the worst case scenarios from Claims 1 and 2, that is, replacing the inequalities for both cases by equalities. Thus, the correlation of the activity patterns found by our SMT-based method provides an upper bound on the correlation of all linear trails compatible with a specified activity pattern for the input and output masks. An optimal trail is shown in Figure 6.9. It spans three rounds and has absolute linear correlation at most $2^{-55.20}$. Therefore, the absolute correlation of all relevant linear trails is bounded by $2^{-55.20}$. It follows that the squared

2-norm of the nontrivial Fourier coefficients of the observed bits $\mathbf{z}$ can be upper bounded by

$$\varepsilon_2 := \|\widehat{p}_{\mathbf{z}} \mathbb{1}_{B_2(2)}\|_2^2 \leq |\operatorname{supp} \widehat{p}_{\mathbf{z}}| \, \|\widehat{p}_{\mathbf{z}} \mathbb{1}_{B_2(2)}\|_\infty^2 \leq 2^{48} \, 2^{-110.40} = 2^{-62.40},$$

where we have used the inequality $|\operatorname{supp} \widehat{p}_{\mathbf{z}}| \leq 2^{48}$. The latter follows from the fact that the observed value $\mathbf{z}$ consists of at most 48 bits in the glitch-extended probing model: if an output coordinate of $\bar{F}$ or $\bar{G}$ is read, at most 24 shares are learned; if an output of the shared linear layer is probed, at most seven shares are observed.

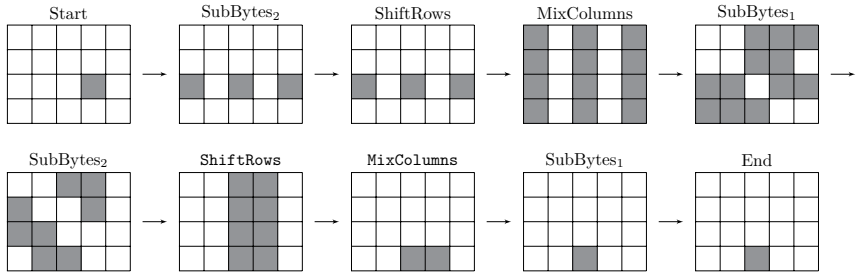


Figure 6.9: An optimal trail (for two probes) with non-zero correlation through the second-order masked AES. The grey cells denote active cells of the output of the operation.

We also built an SMT model for the key schedule. The best trail was found to span eight rounds and activates 21 masked S-boxes. The absolute correlation of the trail is upper bounded by $2^{-63.60}$. Thus, the squared 2-norm of the nontrivial Fourier coefficients of observed bits in the key schedule can be bounded by $\varepsilon_2 \leq 2^{-79.20}$. As the trails through the state transformation have a larger absolute correlation, the upper bound on the maximum advantage of a second-order noisy probing adversary is determined by the bound $\varepsilon_2 \leq 2^{-62.40}$.

Security Claim Due to the analysis in Section 6.5.4, Theorem 6.2 can be applied with the upper bound $\varepsilon_2 \leq 2^{-62.40}$. It follows that the following security claim can be made.

Security Claim 5. Let $\mathcal{A}$ be a noisy 2-threshold probing adversary for the masking of AES. If $\mathcal{A}$ makes at most q queries and the probes of $\mathcal{A}$ are independent and λ-noisy, then the advantage of $\mathcal{A}$ is bounded by (assuming piling-up)

$$\mathrm{Adv}_{\mathsf{noisy\ 2\text{-}thr}}(\mathcal{A}) \leq \sqrt{\frac{q}{\lambda^2 \, 2^{61.4}}} \, .$$

Considering that an adversary can take up to 2^{27} queries, the above bound gives an advantage of $2^{17.2}$ even without noise. Meaning that the noise will only improve the bound. The next section shows that this bound can be significantly improved by taking a closer look at the glitch model, leading to a much lower advantage in practice.

6.5.5 Sharpening the Glitch Model

In Section 6.5.4, an upper bound on the advantage of a second-order noisy probing adversary for the masked AES was obtained. However, we observe that this bound is significantly negatively impacted by the large support of the Fourier transformation of the observed values. Indeed, this leads to an increase in advantage by a factor 2^{24} due to the possibility that a glitch-extended probe may observe 24 bits. In this section, we propose a modified glitch model (supported by simulations) that leads to an improved bound. For ease, we revert back to the regular probing model instead of the noisy probing model from Section 6.2 and remove any noise present on the probed values ($\lambda = 1$). That means we revert back to Theorem 5.2 for the bound on the adversary. Also in our practical experiments, we ran simulations where there was no noise present.

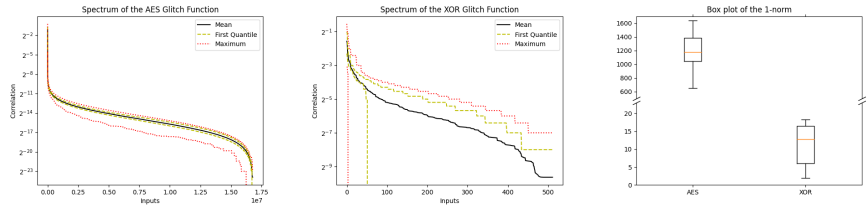
We adapt the glitch-extended probing model from Section 2.5. Instead of an adversary which observes the values of all wires that depend on the probed wire (up to the preceding register stage), the new model proposes that the adversary chooses an arbitrary Boolean function of those values. For example, suppose that the adversary places a glitch-extended probe on a masked function $g(x_1, \ldots, x_n)$. Instead of receiving all of the input values $x_1, \ldots, x_n$, the adversary can (adaptively per query) choose a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and receives the value $f(x_1, \ldots, x_n)$ instead. The function f will be referred to as the *glitch function*.

The idea is that, intuitively, the glitch function cannot be completely arbitrary: most Boolean functions in many variables are not easily realised with a small, structured circuit. We formalise this intuition by postulating that the 1-norm of the Walsh-Hadamard transform of the glitch function can not be too large. To exploit this assumption, the following refinement of Theorem 5.4 is proposed.

Theorem 6.4. *Let $F : \mathbb{V}_a \to \mathbb{V}_b$ be a permutation with $\mathbb{V} \subset \mathbb{F}_2^\ell$ and $I, J \subset \{1, \ldots, \ell\}$. For $\mathbf{x}$ uniform random on $\mathbb{V}_a$ and $\mathbf{y} = F(\mathbf{x})$, let $\mathbf{z} = (g_1(\mathbf{x}_I), g_2(\mathbf{y}_J))$, $f_1 = 2^{-|I|}\,(-1)^{g_1}$, and $f_2 = 2^{-|J|}\,(-1)^{g_2}$. The Fourier transformation of the probability mass function of $\mathbf{z}$ then satisfies*

$$\|\widehat{p}_{\mathbf{z}} - \delta_0\|_2 \leq \|\widehat{f_1}\|_1 \, \|\widehat{f_2}\|_1 \max_{w,w' \in \mathbb{F}_2^\ell/\mathbb{V}^\perp} |C^F_{w,w'}|,$$

where $w, w' \in \mathbb{F}_2^\ell/\mathbb{V}^\perp$ *are such that* $w'_{[\ell]\setminus I} = 0$ *and* $w_{[\ell]\setminus J} = 0$.



(a) AES S-box. (b) 9-bit XOR. (c) 1-norm.

Figure 6.10: Sorted absolute Walsh-spectrum and 1-norm of glitch functions for the AES S-box and 9-bit XOR for different probe positions, times, inputs delays, and initial states.

The above theorem shows that we can improve the bound on a probing adversary if we know the 1-norm of the Walsh-Hadamard transformation of the glitch functions. In order to upper bound this 1-norm, one can simulate the effect of glitches on the circuit.

We simulate the masked AES S-box and compute its Walsh-spectrum for different probe positions. The simulation details can be found in the original work [11]. Since the masked AES S-box has 32 input bits, the memory requirements for performing many experiments are large. To provide an additional example, we also simulate a 9-bit XOR. The spectrum for a single input delay and initial state but different probe positions and time samples for the masked AES S-box is given in Figure 6.10a. The spectrum for different input delays, initial states, probe positions, and time samples for the 9-bit XOR is given in Figure 6.10b. The distributions of the 1-norms are illustrated in Figure 6.10c using a box plot.

Using the observed 1-norm and the above theorem, we can improve the security bound for the masked AES. From Figure 6.10c, we observe that the 1-norm of the Walsh-Hadamard transform of the glitch functions for the masked AES S-box is between 600 and $1635 \approx 2^{10.68}$. By applying Theorem 6.4, the squared 2-norm of the nontrivial Fourier coefficients of the observed bits $\mathbf{z}$ can be upper bounded by

$$\|\widehat{p}_{\mathbf{z}} - \delta_0\|_2^2 \leq 2^{21.36}\, 2^{-110.40} = 2^{-89.04},$$

This gives the following refined security claim.

Security Claim 6. For the second-order masked AES, the following bound on the advantage of the adversary (assuming piling-up) in the probing model

with refined glitches is claimed:

$$\mathrm{Adv}_{2\text{-thr}}(\mathcal{A}) \leq \sqrt{\frac{q}{2^{88}}}\,.$$

Given a maximum of 2^{27} queries, the above bound gives a maximal advantage of $2^{-30.54}$.

6.5.6 Amortising Randomness

Similar to the sharing of the LED in Section 6.4.2, we will extract parts of the state of the masked AES in order to amortise randomness over multiple queries. As opposed to the application to LED which has a total of 48 rounds, AES only has 10 rounds making it more difficult to extract the state multiple times. In order to investigate the security, we again make use of SMT solvers.

The technique works by extracting the randomness from the masked state in certain rounds. Here we extract three out of four shares of each bit of the state for subsequent use. A total of twice the shared state size, 960 random bits, is extracted per masked cipher call. This process is depicted in Figure 6.11.

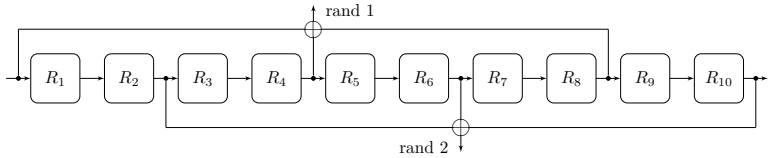


Figure 6.11: Extraction of randomness from a masked AES to create the random bits "rand 1" and "rand 2". The ith round function of AES is denoted by R_i.

Using SMT-based search tools, we find the best linear trail with two probes in one AES circuit and an arbitrary non-zero mask on the extracted randomness. Optimal trails are shown in the original work [11]. The best trail with a non-zero mask on the extracted randomness has absolute correlation at most $2^{-61.0}$. Thus, the dominant trails are still those given in Section 6.5.4 and we again have the bound $\varepsilon_2 \leq 2^{-62.40}$.

Security Claim 7. For the masked AES with randomness reuse, the following bound on the advantage of the adversary (assuming piling-up) in the noisy probing model is claimed:

$$\mathrm{Adv}_{l\text{-blk,noisy 2-thr}}(\mathcal{A}) \leq \sqrt{2l-1} \times \sqrt{\frac{q}{\lambda^2 2^{-61.4}}}\,.$$

By making use of the sharpened glitch model from Section 6.5.5, the above bound can again be improved significantly.

This technique generates a total of 960 random bits per masked cipher call. In total, the masked AES requires 1800 random bits, 936 bits for the state and 864 bits for the key. By choosing $l > 936$, the effective number of random bits for the state is less than one per call. This increases the maximum advantage of a second-order probing adversaries by a factor less than 2^6.

6.6 Conclusion

In this chapter, we had a look at several improvements to the analysis from Chapter 5 and investigated how to create efficient maskings on which the analysis techniques can be used. We found that we can ease the requirement of second-order non-completeness and uniformity in one stage, and instead work with fewer shares. We also improved the latency of our designs by removing register stages. We found it possible to use SMT based tools to speed up the verification process of the needed activity patterns for the bounded-query probing analysis. We found two ways to improve the bound given by Theorem 5.2 such that the effect of glitches can be reduced and that noise can be included. Finally, we discussed a technique to generate randomness using the masking. This technique allows for a further reduction of fresh randomness and for LED we were able to generate as much randomness as it requires.

Nevertheless, even though we are capable of making efficient second-order sharings of lightweight primitives, our masking of AES is still inefficient with a masked S-box requiring over 100k GE. This shows the engineering challenge to provide an efficient second-order AES which has low randomness requirements. Over time, we believe good maskings of the AES including the randomness cost will be found.

Chapter 7

Resilient Uniformity

In this chapter, we generalise the notion of uniformity to a notion resembling Boolean resiliency. We then use this notion to prove the security of the masked construction by Reparaz *et al.* [88].

The entire chapter is written by myself and contains novel research. The writing is based on a work published at Cryptography and Communications where I was supervised by Svetla Nikova. The preprint can be found online [51].

7.1 Motivation

Recall the historical context from Chapter 5: Reparaz *et al.* [88] proposed a method to secure threshold implementations against higher order attacks. The method, called *Consolidating Masking Schemes* (CMS), made use of fresh randomness in order to guarantee the security and even then it was shown that their third-order variant was insecure. The security of their second-order variant was also only argued rather than proven.

In this chapter, we generalise the notion of non-completeness and uniformity such that if masked functions have this notion they are second-order probing secure. The notion resembles the extension from balancedness to resiliency. Unlike the security analysis proposed in Chapter 5, this notion still requires fresh randomness to guarantee the security of a masking. However, it allows us to frame the CMS technique in the world of threshold implementations as being particular examples of *resilient* masked functions. In turn, this insight can reduce the randomness cost the technique requires to provide security.

7.2 Adversary Structures and Resiliency

The generalised notion of uniformity makes use of adversary structures and resembles the definition of Boolean resiliency. We introduce these concept in this section.

Adversary Structures Consider a secret x is shared in s shares $X = \{x^1, ..., x^s\}$. Let us denote all subsets of X by $P(X)$. We call the groups of shares which do not reconstruct the secret as unqualified. The set of unqualified groups, also called the privacy structure, is denoted by $\Delta \subset P(X)$. Let us denote Δ^+ as the maximal set of Δ, *i.e.* the elements in Δ for which no proper superset is also in Δ. In this work, we consider Boolean masking for which Δ consists of all sets containing $s-1$ or fewer shares. For example, consider a secret x is shared into three shares x^1, x^2, x^3, such that $\sum_{i=1}^{3} x^i = x$ then

$$\Delta^+ = \{\{x^1, x^2\}, \{x^2, x^3\}, \{x^1, x^3\}\}.$$

In case multiple secrets are considered, the privacy structure would consist of all sets containing $s-1$ or fewer shares of each secret. For example, for two secrets x, y shared into two shares

$$\Delta^+ = \{\{x^1, y^1\}, \{x^1, y^2\}, \{x^2, y^1\}, \{x^2, y^2\}\}.$$

We denote the *adversary structure* $\Delta_{\mathcal{A}}$ as a monotone decreasing subset of Δ, *i.e.* for all sets in $\Delta_{\mathcal{A}}$ each of their subsets is also in $\Delta_{\mathcal{A}}$. Intuitively, in this work the adversary structure will contain all possible sets of shares a probing adversary can view using a single probe. Conversely, using a probe, an adversary can thus see one element from Δ.

Hirt and Maurer introduced the notion of $\mathcal{Q}^\ell$ adversary structures [66]. An adversary structure $\Delta_{\mathcal{A}}$ is $\mathcal{Q}^\ell$ if no ℓ sets in $\Delta_{\mathcal{A}}$ cover all shares of a secret. In this work, a focus is put on second-order probing security, as such we will work with $\mathcal{Q}^2$ adversary structures.

Resiliency We introduce resiliency as first given by Siegenthaler [96].

A balanced Boolean function is one which gives an output such that each output vector is equally likely to occur if given a random input. In particular, permutations are balanced. The notion of balancedness is then generalised to resiliency which we give in terms of monotone decreasing sets Δ following the work by Braeken *et al.* [21].

Definition 7.1 (Δ resilient [21])**.** Let $f(x)$ be a Boolean function and Δ be a monotone decreasing set. Then $f(x)$ is called Δ resilient if any of its restrictions obtained by fixing an input set $A \in \Delta$ of inputs is balanced.

7.3 Resilient Uniformity

We now introduce the notion of resilient uniformity which provides second-order probing security of threshold implementations. While this section expands the notions of non-completeness and uniformity, the next subsection proves the second-order probing security using these notions. In Section 7.4 the notion of resilient uniformity is applied to the "Consolidating Masking Schemes" (CMS) approach [88].

We start by generalising the notion of non-completeness. Recall that the original notion requires that each share of a function can only work on a limited number of shares namely that each share of a function can not work on all shares of a secret. By using adversary structures as introduced in Section 7.2, we give a more generalised notion of non-completeness by tightening this limitation.

Definition 7.2 (Δ non-completeness)**.** A function $\bar{F}$ is Δ non-complete, for an adversary structure Δ, if each share of $\bar{F}$ only uses inputs A for an $A \in \Delta$.

Thus, the above notion states that each share of the function can only work on a set of input shares specified by the adversary structure Δ. We provide an example of Δ non-completeness. Consider the three-sharing of the multiplication $F(x, y) = xy$

$$F^i = x^i y^i + x^i y^{i+1} + x^{i+1} y^i,$$

where the convention is used that superscripts wrap around at three. The above function is Δ non-complete with

$$\Delta^+ = \{\{x^1, x^2, y^1, y^2\}, \{x^2, x^3, y^2, y^3\}, \{x^1, x^3, y^1, y^3\}\}.$$

In particular, the above Δ is a $\mathcal{Q}^1$ adversary structure as probing two shares of $\bar{F}$ gives back all input shares. Equivalently, taking two elements in Δ^+ covers all shares of a secret, in this case both x and y.

It is clear that if Δ is a $\mathcal{Q}^d$ adversary structure then a Δ non-complete function is also d^{th}-order non-complete.

Now, assume that each shared function in the threshold implementation is Δ non-complete. Looking back at the notion of resiliency from Section 7.2, we

transform the notion of uniformity to its resilient equivalent. A first attempt in this transformation would give a notion where fixing certain inputs to constants again gives a uniform function thus ensuring that the output is still uniform even after one probe was placed. Nevertheless, this simple adaptation is not sufficient. As a glitch-extended probe gives back all input shares of a share of a function, the output of a probed function can never be uniform. An adaptation to the notion of resiliency is necessary.

We denote a *restriction* of a masking $\bar{x}$ on a set of shares A by $\bar{x}_A$ to be the set of shares of $\bar{x}$ indicated by A. For example, for a three-sharing $\bar{x} = (x^1, x^2, x^3) = (0, 1, 0)$ and $A = \{x^1, x^2\}$ then $\bar{x}_A = (0, 1)$. We start with the definition of a Δ resilient uniform masking.

Definition 7.3 (Δ resilient uniform masking)**.** A masking of a variable x is Δ resilient uniform if there is a set $I \in \Delta$, such that when $\bar{x}_I$ are fixed to constants then $\bar{x}_{X \setminus I}$ is uniform, with X the set of all shares.

For example, taking a three-shared Boolean masking of a secret $x \in \mathbb{F}_2$, $Sh(x) = \{(x^1, x^2, x^3) \mid \sum_{i=1}^{3} x^i = x\}$. Considering Δ as the set of sets containing at most one share, *i.e.* $\Delta = \{\{x^1\}, \{x^2\}, \{x^3\}, \emptyset\}$. The sharing with one share set to zero would be a Δ resilient uniform masking. In other words, $\{(0, x^2, x^3) \mid x^2 + x^3 = x\}$ would no longer be uniform, but it is a Δ resilient uniform masking.

Having defined what is a Δ resilient uniform masking, a (Δ, Δ') resilient uniform function is a function which maps a Δ resilient input masking to a Δ' resilient output masking. The definition of a (Δ, Δ') resilient uniform function is given in its combinatorial form.

Definition 7.4 ((Δ, Δ') resilient uniformity)**.** A function $\bar{F}(\bar{x}) = \bar{y}$ is (Δ, Δ') resilient uniform if $\forall I \in \Delta$, $\exists J \in \Delta'$, such that $\forall \bar{x}^*_I \in \mathbb{F}_2^{|I|}$, $\bar{y}^*_J \in \mathbb{F}_2^{|J|}$, $\exists c \in \mathbb{N}$, such that $\forall x \in \mathbb{F}_2^n$ and $\bar{y} \in Sh(F(x))$ with $\bar{y}_J = \bar{y}^*_J$:

$$\left| \left\{ \bar{x} \in Sh(x) \text{ with } \bar{x}_I = \bar{x}^*_I \,\middle|\, \bar{F}(\bar{x}) = \bar{y} \right\} \right| = c\,.$$

Later on, in Section 7.4 we will see that each second-order non-complete function that is re-masked via the ring refreshing method achieves a notion of (Δ, Δ') resilient uniformity.

7.3.1 Proving Second-Order Probing Security

Having the notion of resilient uniformity, we prove that it is sufficient for perfect second-order probing security of threshold implementations as defined

in Section 2.6.1. We start by showing that a (Δ, Δ') resilient uniform function indeed maps a Δ resilient uniform input masking to a Δ' resilient uniform output masking.

Lemma 7.1. *A (Δ, Δ') resilient uniform function maps a Δ resilient uniform input masking to a Δ' resilient uniform output masking.*

Proof. Consider a random input masking $\bar{x}$ of the secret x which is Δ resilient uniform. Meaning there is a set $I \in \Delta$, such that $\bar{x}_I \in \mathbb{F}_2^{|I|}$ are fixed to constants, denote these by $\bar{x}_I^*$, and $\bar{x}_{X \setminus I}$ is uniform. Also consider $\bar{F}$ a (Δ, Δ') resilient uniform function. We show that there exists a set $J \in \Delta'$, such that the output $\bar{y} = \bar{F}(\bar{x})$ is a Δ' resilient uniform masking for that J.

From the definition of (Δ, Δ') resilient uniformity, we know that there exists a set $J \in \Delta'$, such that for all $\bar{y}_J^* \in \mathbb{F}_2^{|J|}$, there exists a $c \in \mathbb{N}$, such that $\forall \bar{y} \in Sh(F(x))$ with $\bar{y}_J = \bar{y}_J^*$:

$$\left| \left\{ \bar{x} \in Sh(x) \text{ with } \bar{x}_I = \bar{x}_I^* \,\middle|\, \bar{F}(\bar{x}) = \bar{y} \right\} \right| = c \,.$$

Since $\bar{F}$ is a function, there exists at least one vector of constants $\bar{y}_J^*$, such that $c \neq 0$. We claim that $\bar{y}$ is Δ' resilient uniform for $J \in \Delta'$ and the previous mentioned set of constants. Indeed, each instance $\bar{y}$ with $\bar{y}_J = \bar{y}_J^*$ occurs equally frequently, namely with frequency $c \neq 0$. As a result $\bar{y}_{Y \setminus J}$ is uniform and thus $\bar{y}$ is Δ' resilient uniform. □

We then show that the notion of resilient uniformity is stronger than regular uniformity.

Lemma 7.2. *A (Δ, Δ') resilient uniform function $\bar{F}$ is uniform.*

Proof. We take an arbitrary $I \in \Delta$ and an arbitrary $\bar{x}_I^* \in \mathbb{F}_2^{|I|}$. We know there is a $J \in \Delta'$, such that for all $\bar{y}_J^* \in \mathbb{F}_2^{|J|}$, there is a constant c, such that for all x and $\bar{y} \in Sh(F(x))$ with $\bar{y}_J = \bar{y}_J^*$:

$$|\{\bar{x} \in Sh(x) \text{ with } \bar{x}_I = \bar{x}_I^* \,|\, \bar{F}(\bar{x}) = \bar{y}\}| = c \,.$$

We pick a $\bar{y}_J^* \in \mathbb{F}_2^{|J|}$ such that c does not equal to zero.

We now take another $\bar{x}_I' \in \mathbb{F}_2^{|I|}$, then we find that there is another constant c', such that for all x and $\bar{y} \in Sh(F(x))$ with $\bar{y}_J = \bar{y}_J^*$:

$$|\{\bar{x} \in Sh(x) \text{ with } \bar{x}_I = \bar{x}_I' \,|\, \bar{F}(\bar{x}) = \bar{y}\}| = c' \,.$$

By taking the sum of these two we find that for all x and $\bar{y} \in Sh(F(x))$ with $\bar{y}_J = \bar{y}^*_J$:

$$|\{\bar{x} \in Sh(x) \text{ with } \bar{x}_I = \bar{x}^*_I \text{ or } \bar{x}_I = \bar{x}'_I \,|\, \bar{F}(\bar{x}) = \bar{y}\}| = c + c' \,.$$

Thus, by looping over all possible $\bar{x}_I$, we find that there is a constant c^*, such that for all x and $\bar{y} \in Sh(F(x))$ with $\bar{y}_J = \bar{y}^*_J$:

$$|\{\bar{x} \in Sh(x) \,|\, \bar{F}(\bar{x}) = \bar{y}\}| = c^* \,.$$

Since each share in $\bar{y}$ and $\bar{x}$ is fixed, c^* needs to equal $2^{n(s_x-1)}/2^{m(s_y-1)}$ if $x \in \mathbb{F}_2^n$ with s_x input shares and $y \in \mathbb{F}_2^m$ with s_y output shares. Since the constant c^* has the same value for each choice of secret x and $\bar{y}^*_J$, we have shown the uniformity of $\bar{F}$. □

There is also a link between resilient uniform functions of different adversary structure. Namely a (Δ, Δ') resilient uniform function is (Δ, Δ'') resilient uniform when $\Delta' \subset \Delta''$.

Lemma 7.3. *Given two adversary structures $\Delta' \subset \Delta''$, a (Δ, Δ') resilient uniform function is also (Δ, Δ'') resilient uniform.*

Proof. Let $\bar{F}$ be a (Δ, Δ') resilient uniform function. We prove that $\bar{F}$ is (Δ, Δ'') resilient uniform. Take an arbitrary $I \in \Delta$, then from the (Δ, Δ') resilient uniformity, there there exists a $J \in \Delta'$, such that for all $\bar{x}^*_I \in \mathbb{F}_2^{|I|}$ and all $\bar{y}^*_J \in \mathbb{F}_2^{|J|}$ there exists a constant c, such that for all x and $\bar{y} \in Sh(F(x))$ with $\bar{y}_J = \bar{y}^*_J$:

$$\left| \left\{ \bar{x} \in Sh(x) \text{ with } \bar{x}_I = \bar{x}^*_I \,\middle|\, \bar{F}(\bar{x}) = \bar{y} \right\} \right| = c \,.$$

However, since $\Delta' \subset \Delta''$, J is also an element of Δ''. Thus, we can always find a suitable $\bar{y}^*_{J'}$ with $J' \in \Delta''$ for which the necessary property holds. □

Finally, we show that the composition between resilient uniform functions is again resilient uniform.

Lemma 7.4. *The composition between a (Δ, Δ') resilient uniform function and a (Δ', Δ'') resilient uniform functions is (Δ, Δ'') resilient uniform, given that the functions can compose.*

Proof. Let $\bar{F}$ be a (Δ', Δ'') resilient uniform function and $\bar{G}$ a (Δ, Δ') resilient uniform function such that the composition $\bar{F} \circ \bar{G}$ makes sense, we prove that $\bar{F} \circ \bar{G}$ is (Δ, Δ'') resilient uniform. We take an arbitrary $I \in \Delta$. From $\bar{G}$ being (Δ, Δ') resilient uniform, we know that there exists a $J \in \Delta'$, such that for all

$\bar{x}_I^* \in \mathbb{F}_2^{|I|}$ and all $\bar{y}_J^* \in \mathbb{F}_2^{|J|}$ there exists a constant c, such that for all x and $\bar{y} \in Sh(G(x))$ with $\bar{y}_J = \bar{y}_J^*$:

$$\left| \left\{ \bar{x} \in Sh(x) \text{ with } \bar{x}_I = \bar{x}_I^* \,\middle|\, \bar{G}(\bar{x}) = \bar{y} \right\} \right| = c\,.$$

From $\bar{F}$ being (Δ', Δ'') resilient uniform we also know that there exists a $K \in \Delta''$, such that for all $\bar{y}_J^* \in \mathbb{F}_2^{|J|}$ and $\bar{z}_K^* \in \mathbb{F}_2^{|K|}$ there exists a constant c', such that for all y and $\bar{z} \in Sh(F(y))$ with $\bar{z}_K = \bar{z}_K^*$:

$$\left| \left\{ \bar{y} \in Sh(y) \text{ with } \bar{y}_J = \bar{y}_J^* \,\middle|\, \bar{F}(\bar{y}) = \bar{z} \right\} \right| = c'\,.$$

Thus, we find that there are a constant number of $\bar{y}$ vectors with $\bar{y}_J = \bar{y}_J^*$ which are mapped to a vector $\bar{z}$ with $\bar{z}_K = \bar{z}_K^*$. For each of these $\bar{y}$ vectors we find a constant number of $\bar{x}$ vectors with $\bar{x}_I = \bar{x}_I^*$ that are mapped to that $\bar{y}$. Thus, we find that there exists a set K, such that for all $\bar{x}_I^* \in \mathbb{F}_2^{|I|}$ and $\bar{z}_K^* \in \mathbb{F}_2^{|K|}$ there exists a constant c^*, namely $c^* = cc'$ or $c^* = 0$, such that for all x and $\bar{z} \in Sh(F(G(x)))$ with $\bar{z}_K = \bar{z}_K^*$:

$$\left| \left\{ \bar{x} \in Sh(x) \text{ with } \bar{x}_I = \bar{x}_I^* \,\middle|\, \bar{F}(\bar{G}(\bar{x})) = \bar{z} \right\} \right| = c^*\,.$$

Since I was chosen arbitrarily, we have proven the lemma. □

The above two lemmas form a compositional security argument effectively showing that (Δ, Δ') resilient uniformity is sequentially composable. We now show that the notion of (Δ, Δ') resilient uniformity leads to second-order probing security. We require that each function in the circuit is $(\Delta_i, \Delta'_{i+1})$ resilient uniform such that $\Delta'_i \subset \Delta_i$ with i the index enumerating the shared functions in the circuit.

Theorem 7.5. *The composition of shared functions $\bar{F}_i$ which are correct, Δ_i non-complete, and $(\Delta_i, \Delta'_{i+1})$ resilient uniform, with each Δ_i, Δ'_i being $\mathcal{Q}^2$ adversary structures and $\Delta'_i \subset \Delta_i$, is secure against a second-order probing adversary.*

Proof. Due to all functions being correct, the circuit is also correct. We show the circuit is also second-order private as defined in Section 2.6.1. Recall that privacy is shown as the existence of a simulator who can simulate the probed values from scratch. Take two arbitrary probed functions from the threshold circuit and give the adversary all the inputs of those functions. We denote the set of shares given to the adversary by $\mathcal{I}$.

We treat the case where the two probed functions lie in the same shared function $\bar{F}_i$. Due to the Δ_i non-completeness of $\bar{F}_i$ and since Δ_i is a $\mathcal{Q}^2$ adversary structure, $\mathcal{I}$ never contains all shares of an input. The security of

this case relies only on the original definition of uniformity. Since each shared function is also uniform, the input masking was also uniform, thus each input share can be simulated as uniformly random.

We look at the case where the two probed functions do not lie in the same shared function. Denote the to probed functions $\bar{F}_i$ and $\bar{F}_j$ with $i \neq j$. Due to the Δ_i and Δ_j non-completeness of both functions, $\mathcal{I}$ consists of a set of shares $\bar{x}_I$ with $I \in \Delta_i$ and a set of shares $\bar{y}_J$ with $J \in \Delta_j$. Since the first probed function has a uniform input, the shares $\bar{x}_I$ can be simulated as uniform randomness. Consider the function $\bar{F}$ which maps the input of $\bar{F}_i$ to the input of $\bar{F}_j$.

We first prove that $\bar{F}$ is (Δ_i, Δ'_j) resilient uniform. We know that $\bar{F}$ is the composition of functions which are $(\Delta_k, \Delta'_{k+1})$ resilient uniform, for $k \in \{i, ..., j-1\}$. From Lemma 7.3, we know from each $\Delta'_k \subset \Delta_k$, that each function is also (Δ_k, Δ_{k+1}) resilient uniform. From Lemma 7.4, we then know that the composition of these functions is (Δ_i, Δ_j) resilient uniform. Thus, $\bar{F}$ is (Δ_i, Δ_j) resilient uniform.

As a result, since the input of $\bar{F}$ is a Δ_i resilient uniform masking, the output of $\bar{F}$ is also a Δ_j resilient uniform masking following Lemma 7.1. Thus, the output of $\bar{F}$ consists of constants and a uniform masking. Since Δ_j is a $\mathcal{Q}^2$ adversary structure, the resulting uniform masking is still non-complete. Thus, the second probe only gives back constants, which are already simulated, or uniform randomness. As a result, all probed values can be simulated from scratch. □

In case functions are only uniform but not (Δ, Δ') resilient, one can find examples such that, when composed, the functions become insecure against a second-order attack. An example is given by Reparaz [87] of a shared identity function using five shares.

$$(a^1, a^2, a^3, a^4, a^5) \rightarrow (a^1, a^1 + a^2 + a^4 + a^5, a^1 + a^3 + a^4 + a^5, a^4, a^5)$$

This function is uniform but it is clearly not second-order secure as one probe can read a^3 and the other $a^1 + a^2 + a^4 + a^5$ to reveal a. The above function is not (Δ, Δ') resilient as, for example taking a^3 in the input equal to zero gives the non-uniform output distribution $(a^1, a^1 + a^2 + a^4 + a^5, a^1 + a^4 + a^5, a^4, a^5)$. Fixing an output coordinate to a constant does not help make the distribution uniform, for example one can fix the third coordinate to zero ($a^1 + a^4 + a^5 = 0$) and remove it. However, the resulting output distribution $(a^1, a^2, a^4, a^1 + a^4)$ is again non-uniform. This will be the case for any other output coordinate one fixes.

7.4 Consolidating Masking Schemes Revisited

This section applies the notion of resilient uniformity to the "Consolidating Masking Schemes" (CMS) approach [88]. The CMS approach was proposed to secure threshold implementations against d^{th}-order probing adversaries using fresh randomness. However, its security was only argued and formal definitions were never given. Later on, Moos *et al.* [77] showed that the CMS approach was insecure against third- or higher-order attacks showing a need for formal security proofs. In this work, we prove the second-order security of the CMS approach.

7.4.1 The CMS Approach

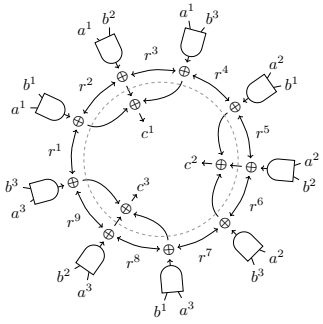
We quickly introduce the CMS approach which guarantees second-order probing security. In short, the approach takes a second-order non-complete map and refreshes the uniformity by adding extra randomness in a ring refreshing approach. The expanded shares are then re-compressed by XORing them together.

We give an example of the approach, namely for a shared AND gate which is depicted in Figure 7.1a. First, the CMS approach calculates all the linear and nonlinear terms, of the shared function in this case all cross products $a^i b^j$. These terms are then added together, such that second-order non-completeness still holds, *i.e.* the joint inputs of every pair of functions do not cover all shares of a secret. In the AND gate example, each cross product is held separately as XORing some of them would already invalidate second-order non-completeness. In order to re-compress the shares, they are first added with randomness and synchronised. The addition of randomness follows a *ring refreshing* approach as shown in Figure 7.1b with the dashed line denoting a register stage.

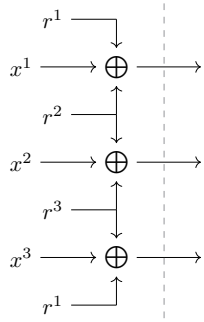
While the example is that of a multiplication, the CMS approach can be applied to any Boolean function.

7.4.2 Applying Resilient Uniformity

We split up the CMS approach in two main layers; An expansion layer which expands the input shares and adds randomness; and a compression layer re-compressing the shares. The two layers are depicted in Figure 7.2. We show that both layers conform to the notion of resilient uniformity.



(a) A second-order CMS-secured AND-gate.



(b) Three-shared ring refreshing.

Figure 7.1: The CMS secure AND gate and refreshing method.

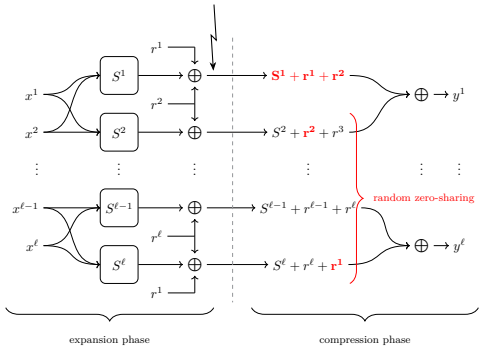


Figure 7.2: A depiction of the CMS approach. The gray dashed line denotes a register stage. The adversary probes the first expanded share. Values known to the adversary are denoted in bold red.

To that end we introduce the two main adversary structures. The first is denoted by Δ_{one_share} which is the adversary structure consisting of all sets with one share, in total, and the empty set. Second, for a shared function $\bar{F}$, we denote the natural adversary structure given by all sets of shares a single probe can observe by $\Delta_{probing}$. For example, consider the function $G(x, y, z) = xy + z$ which is shared in the following way

$$G^i = x^i y^i + x^i y^{i+1} + x^{i+1} y^i + z^i,$$

where the convention is used that superscripts wrap around at three. Then Δ_{one_share} and $\Delta^{+}_{probing}$ equal the following sets

$$\Delta_{one_share} = \{\{x^1\},\{x^2\},\{x^3\},\{y^1\},\{y^2\},\{y^3\},\{z^1\},\{z^2\},\{z^3\},\emptyset\},$$

$$\Delta^{+}_{probing} = \{\{x^1,x^2,y^1,y^2,z^1\},\{x^2,x^3,y^2,y^3,z^2\},\{x^1,x^3,y^1,y^3,z^3\}\}.$$

We claim that if a function is shared according to the CMS approach, it is $(\Delta_{probing}, \Delta_{one_share})$ resilient uniform.

We show the expansion phase of the CMS is $(\Delta_{probing}, \Delta_{one_share})$ resilient uniform. This expansion phase $\bar{F}$ is a $\mathbb{F}_2^{m+n} \to \mathbb{F}_2^m$ function with inputs $(a^1,...,a^n,r_1,...,r_m)$ given by the equations

$$F^i = G^i(a^1,...,a^n) \oplus r_i \oplus r_{i+1}\,,$$

for a second-order non-complete function $\bar{G}$, $i \in \{1,...,m\}$ and where the subscripts wrap around at m.

Theorem 7.6. *The function $\bar{F}$ as defined above is $(\Delta_{probing}, \Delta_{one_share})$ resilient uniform.*

Proof. We take an arbitrary set I from $\Delta_{probing}$. Without loss of generality we assume this set consists of the elements $\{r_i, r_{i+1}\}$ for some $i \in \{1,...,m\}$ and a non-complete set of shares of $\bar{a}$, *i.e.* a set which misses at least one share of $\bar{a}$, which we denote by A. We take $J = \{F^i\}$, set $\{A, r_i, r_{i+1}\}$ to arbitrary constants $\bar{a}^*_A \in \mathbb{F}_2^{|A|}, r^*_i \in \mathbb{F}_2, r^*_{i+1} \in \mathbb{F}_2$, and set $F^i = G^i(a^1,...,a^n) \oplus r_i \oplus r_{i+1}$. Setting F^i to a different value would make the proof evident for $c = 0$. We show that there exists a constant c, such that for an arbitrary a and $\bar{y} \in Sh(F(a))$ where $F^i = y^i = G^i(a^1,...,a^n) \oplus r_i \oplus r_{i+1}$, there are c inputs $(a^1,...,a^n,r_1,...,r_m)$, adhering to the constraints $\bar{a}_A = \bar{a}^*_A, r_i = r^*_i, r_{i+1} = r^*_{i+1}$, mapping to $\bar{y}$.

Since A consists of a non-complete set of shares, we find that for each secret a there are a constant number, denote this c', of share vectors $\bar{a} \in Sh(a)$ with $\bar{a}_A = \bar{a}^*_A$. We now take an arbitrary vector of shares $\bar{a}$ from the c' previous possibilities. We show that for each such vector we find unique values $\{r_1,...,r_m\}$, such that $(a^1,...,a^n,r_1,...,r_m)$ maps to $\bar{y}$. This is evident, since the shares $\{a^1,...,a^n\}$ are already fixed, we set

$$r_j \oplus r_{j+1} = y^j \oplus G^j(a^1,...,a^n)\,.$$

The above gives $m-1$ linear equations with $m-2$ variables (excluding the equation for F^i and removing the variables r_i, r_{i+1}). Remark that due to the correctness property, one equation is the sum of all others. Removing that linear dependent equation, we have $m-2$ equations and $m-2$ variables. It

is clear that one can assign a unique r_j to each equation, thus each remaining equation is linear independent and there is a unique solution for $\{r_1, ..., r_m\}$. As a result, we find that $c = c'$ and thus for each a and $\bar{y} \in Sh(F(a))$ where $y^i = G^i(a^1, ..., a^n) \oplus r_i \oplus r_{i+1}$ we find there are c' $\Delta_{probing}$-restricted inputs which map to $\bar{y}$. □

Thus, the expansion phase is $(\Delta_{probing}, \Delta_{one_share})$ resilient uniform. The second phase, the compression phase, maps each expanded share to only one output. As a result, the inputs viewed by a probe are only used for one output share and thus a $\Delta_{probing}$ resilient input gets mapped to a Δ_{one_share} resilient output meaning that the second phase is $(\Delta_{probing}, \Delta_{one_share})$ resilient uniform.

Since both layers adhere to the same notion of resilient uniformity, the entire CMS approach is $(\Delta_{probing}, \Delta_{one_share})$ resilient uniform. Since $\Delta_{one_share} \subset \Delta_{probing}$, from Theorem 7.5 we know that the approach offers second-order probing security.

7.5 Conclusion

In this chapter, we discussed a generalisation of the non-completeness and uniformity notions. These *resilient* notions provide protection against higher-order attacks and allow us to better frame works like the CMS approach in the world of threshold implementations and Boolean functions.

As shown in the original work [51], the notion can also be used to reduce the randomness cost of CMS-masked solutions. This is particularly useful for primitives such as PRESENT, meaning primitives with diffusion layers that do not allow for long zero-correlation trails or where the confusion of the masked S-boxes is not high enough.

Chapter 8

Threshold Implementations Against Advanced Leakage

In this chapter, we apply the security analysis from Chapter 5 to show maskings are secure against transition and coupling leakage. We take black-box first-order glitch-extended secure maskings of several symmetric primitives, analyse them, and add randomness in order to secure them against the more advanced leakage.

The entire chapter is based on a work written by myself and accepted at Indocrypt. You can find the preprint online [48].

8.1 Analysing Transition Leakage

Since the security analysis from Chapter 5 can easily be performed cipher-wide, it becomes easier to analyse the security of a masking against more advanced effects such as transition leakage. Recall from Section 2.5.4, that via transition leakage, the adversary can view the digital values on hardware during two different cycles. Namely, the adversary can observe the difference between the new and old digital values that flowed through the resource. In symmetric primitives, this often translates to an adversary being able to observe two masked S-boxes using only one probe. The architecture of the masking determines which two S-boxes are observed using a probe.

Finally, we consider the combined effect of glitches and transition leakage. According to Faust *et al.* [59] probing a memory gate is equivalent to probe the

sole input (i.e. the wire) to it which can be considered also as output of the particular simple logic which ends up with the considered memory gate. In that regard, leakage caused by glitches might seem stronger than the leakage caused by transitions. However, in practice the two leakages can manifest differently in the time interval representing a single cycle, namely the glitches and the logic transitions will occur in the beginning while the memory transition will occur only at the end. As illustrated by Faust *et al.*, there could be a time window between the computational and the storage phases.

We apply the extended probing model to arbitrary glitch-extended probing secure masking of several architectures of PRESENT, AES, and ASCON to investigate how to best protect the masked primitives. In our analysis we consider a black box masking of the S-box to make the analysis more general. We just assume the linear layers are masked share-wise and that the masked S-boxes do not share inputs such as recycled randomness.

8.1.1 PRESENT

We recall the PRESENT cipher from the work of Bogdanov *et al.* [18]. The input to PRESENT is a 64-bit plaintext m. Each round comprises an XOR with the round key, a substitution layer, and a permutation layer. The substitution layer consists of 16 applications of a four-bit cubic S-box. The permutation layer of PRESENT is a bit permutation which is depicted in Figure 8.1. The following arguments are also applicable to the GIFT cipher [3].

Round-Based Architectures Require Extra Protection We first consider a glitch-extended probing secure masking (such as a threshold implementation) in a round-based architecture. In such an architecture each masked S-box in the round function is implemented separately on the platform. When an adversary places a transition-extended probe in an S-box of a round-based architecture, it can view the computation of the same implemented S-box in two consecutive rounds. In this architecture, the diffusion layer of PRESENT allows for some weak points concerning transition leakage. More specifically, bits $0, 21, 42$ and 63 are mapped to the same position. As a result, transition leakage from the $0, 5, 10$ or 15th masked S-box could reveal information. An example activity pattern is indicated in dotted blue in Figure 8.1. The weakness can be resolved by re-masking the bits which remain fixed through the diffusion layer. This randomness can be re-used every round and can be the same over the bits $0, 21, 42$ and 63. For the previously mentioned sharing of the S-box this countermeasure would cost a total of two random bits for the entire masked cipher.

So far we assumed the shared S-box was implemented as a rolled-out circuit on the platform. However, the typical threshold implementation of the PRESENT S-box consists of two degree two maps $\bar{G}$. As an example, consider a decomposition of the PRESENT S-box as follows

$$S(x, y, z, w) = B'(G(G(C'(x, y, z, w) + d)) + e)\,.$$

In the above, the nonlinear function $G(x, y, z, w)$ is given as

$$G_1 = x + yz + yw \qquad G_2 = w + xy \qquad G_3 = y \qquad G_4 = z + yw\,,$$

the linear transformations as

$$B' = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}, C' = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix},$$

and the constants as

$$d = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}, e = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}.$$

The permutation $G(x, y, z, w)$ is shared using a direct balanced sharing. More specifically, for each share $i \in \{1, 2, 3\}$

$$G_1^i = x^i + y^i z^i + y^i z^{i+1} + y^{i+1} z^i + y^i w^i + y^i w^{i+1} + y^{i+1} w^i\,,$$

$$G_2^i = w^i + x^i y^i + x^i y^{i+1} + x^{i+1} y^i\,,$$

$$G_3^i = y^i\,,$$

$$G_4^i = z^i + y^i w^i + y^i w^{i+1} + y^{i+1} w^i\,,$$

where the convention is used that superscripts wrap around at three.

A designer can implement $\bar{G}$ and evaluate it twice to compute an S-box. We denote the two parts of the shared S-box by $\bar{S}_1$ and $\bar{S}_2$. If an adversary uses a transition-extended probe on the shared S-box it can view the computation of both $\bar{S}_1$ and $\bar{S}_2$. This is depicted in blue in Figure 8.2.

We give an example that the above explained weakness can constitute a probing attack for a particular sharing of the S-box. In particular, we consider the blue activity pattern from Figure 8.2. Considering the sharing of the S-box above. Consider, for a secret x, the shares x^1, x^2, x^3 such that $\sum_{i=1}^{3} x^i = x$.

An adversary placing a glitch and transition-extended probe $\mathcal{P}$ in the first coordinate of $\bar{S}_1(\bar{x}, \bar{y}, \bar{z}, \bar{w})$ is given back the shares $x^1, y^1, y^2, z^1, z^2, w^1$, and w^2. However, from the second part of the shared S-box $\bar{S}_2$ the adversary views the input values (equivalently $\bar{S}_1$'s output values)

$$\bar{S}_1^1 = w^1 + x^1y^1 + x^1y^2 + x^2y^1\,,$$

$$\bar{S}_1^2 = w^2 + x^2y^2 + x^2y^3 + x^3y^2\,.$$

Thus, $\mathcal{P} = \{\bar{S}_1^1, \bar{S}_1^2, x^1, y^1, y^2, z^1, z^2, w^1, w^2\}$. Consider random secrets x, y, z, w, then we find that

$$I(x; \mathcal{P}) = H(x) - H(x|\mathcal{P}) \neq 0,$$

with I the mutual information and H the Shannon entropy. In other words, a glitch- and transition-extended probing adversary can break the design.

As a conclusion, a straightforward round-based architecture of PRESENT could be vulnerable to transition leakage and thus extra costs would be required to secure the architecture.

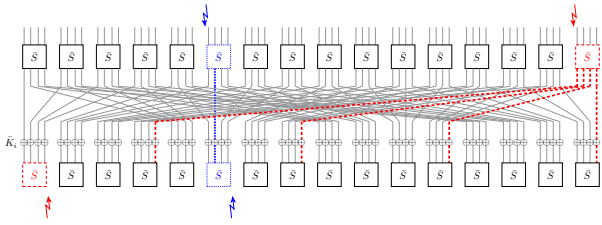


Figure 8.1: An activity pattern caused by transition leakage in PRESENT. In dotted blue lines, we find harmful transition leakage using a round-based architecture. In dashed red lines, we find perfect security using a serial architecture.

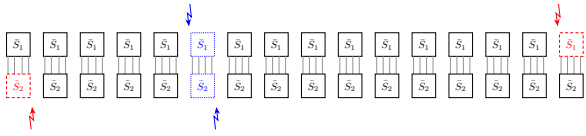


Figure 8.2: Transition leakage in the calculation of the shared S-box divided in $\bar{S}_1$ and $\bar{S}_2$. The dotted blue lines denote leakage in a round-based architecture. The dashed red lines denote leakage in a serial architecture.

Serial Architectures are Secure As a second example, we consider a glitch-extended probing secure masking in an S-box serial architecture. In this architecture only one shared S-box is implemented on the platform and each S-box in the cipher is computed in series. Following Figure 8.1, the architecture computes the S-boxes from left to right. Transition leakage from such a design occurs either between two S-boxes in the same round or between the last and first S-box of two consecutive rounds. Assuming the shared S-boxes are uniform, the former case never constitutes a weakness. The latter case is harmless thanks to the linear layer of PRESENT as indicated in red in Figure 8.1. We see, no matter which input or output mask is chosen, the resulting hull will always be a zero-correlation linear approximation. Thus, no extra precautions need to be made to ensure security against transition leakage.

In case the shared S-box is calculated over two cycles using the same resources, the architecture can be secured against transition leakage by if all $\bar{S}_1$ are first calculated from left to right and before calculating the $\bar{S}_2$. A transition-extended probe either only views two separate $\bar{S}_1$ (similarly $\bar{S}_2$) in the same cycle or it views the parts as depicted in red in Figure 8.2. It is clear that in no case the adversary can break the masking using transition leakage.

Considering architectures of a masked PRESENT in light of transition leakage, we find that serial architectures provide protection without requiring additional costs.

8.1.2 AES

Recall the AES construction as introduced in Section 4.6.1. There are many primitives using a similar diffusion layer such as LED [65], PHOTON [64], PRINCE [19], SKINNY [6], etc. The security analysis considered here applies to all these primitives.

Every Architecture Requires Extra Protection For a glitch-extended probing secure AES, we find that architectures are vulnerable against transition leakage in the application of the `MixColumns` operation. First, notice that no matter the architecture, transition leakage from the computation of an S-box is never usable for an adversary due to the branch number of the `MixColumns`. An example of such an activity pattern is depicted in Figure 8.3. However, during the computation of the `MixColumns` and due to the effect of glitches, the adversary can view the input of the operation and place a mask such that only one output byte is active. This active byte can only shift over the state, causing the adversary to also observe the computation of the following `MixColumns` due

to transition leakages. An activity pattern for a round-based architecture is shown in Figure 8.4. Even by changing the order of the `MixColumns` harmful activity patterns can always be found.

One can prevent this leakage by re-masking some cells (for example, the top row for the round-based architecture) of the AES state. This randomness cost is low as the same randomness can be used for every round and for every cell. More specifically, considering a two-shared threshold implementation, a total of eight random bits are needed to prevent any transition leakage from occurring.

Another countermeasure is based on the verification of the Linear Approximation Table (LAT) of the shared AES S-box using the definitions from Section 5.4. More specifically, note that a masking of a linear operation works share-wise. As a result, a transition-extended probe on the AES diffusion layer only views one share each of two `MixColumns` operations and thus one input and one output share of the shared S-box. Given an s-sharing of the AES S-box $\bar{S}$ and $\mathbb{V} = \bigoplus_{i=1}^{s} \mathbb{V}^i$ such that each mask $u \in \mathbb{F}_2^{8s}/\mathbb{V}^\perp$ can be decomposed in $(u^1, ..., u^s)$ with $u^i \in \mathbb{F}_2^8/(\mathbb{V}^i)^\perp$ then the linear approximation between the observed values is only harmful if

$$\forall \alpha, \beta \in \mathbb{F}_2^{8s}/\mathbb{V}^\perp : \; C^{\bar{S}}_{\alpha,\beta} = 0 \text{ when } \mathrm{wt}(\alpha) = 1, \mathrm{wt}(\beta) = 1 \,.$$

In words, the sharing is secure if the masked S-box has nontrivial diffusion between the shares. A sharing using randomness, for example the sharing of the AES S-box by De Cnudde *et al.* [43], typically has this property. Currently there exists no uniform sharing of the AES S-box and thus every AES S-box sharing (except those using changing of the guards as explained later) uses randomness. However, applying the above property to sharings of primitives such as LED or PRINCE could be interesting future work.

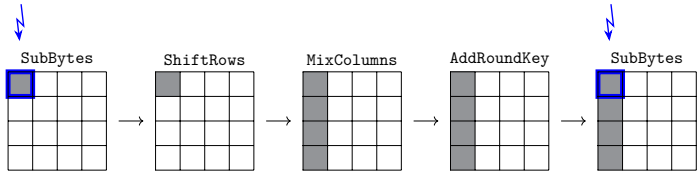


Figure 8.3: Activity patterns between the top left S-boxes of rounds i and $i+1$ of a masked AES. The figure denotes in blue what the adversary can observe and grey cells denote active cells.

Using Changing of the Guards Currently, the AES S-box has no known uniform sharing. As a result, designers typically use the changing of the guards

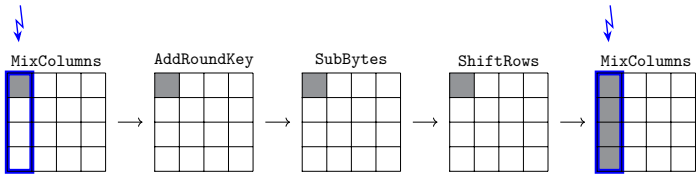


Figure 8.4: An activity pattern caused by transition leakage following the AES diffusion layers. The figure denotes in blue what the adversary can observe and grey cells denote active cells.

technique by Daemen [35] to ensure uniformity, see Section 4.6.2. We already saw in Section 6.5.3 that the changing of the guards method influences the diffusion of the masking.

Here we investigate that if the changing of the guards technique is used with care, one might use it to strengthen the masking against transition leakage. For example, while the diffusion following the left picture still allows for harmful transition leakage in a round-based architecture, the diffusion following the second picture prevents the adversary from learning a secret variable. This is due to the pattern ensuring that each active cell in the state activates at least one different column after the application of the `SubBytes` and `ShiftRows` operations. The third picture of Figure 8.5 depicts which column each cell activates after the `SubBytes` and `ShiftRows` operations.

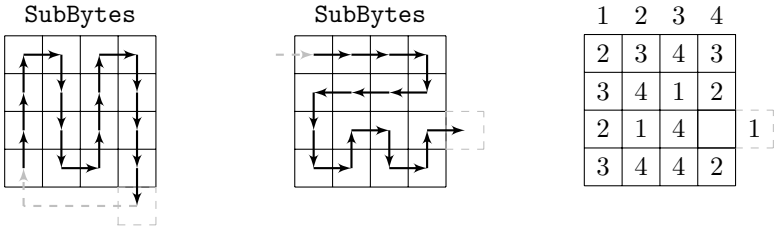


Figure 8.5: Two example diffusion patterns using the changing of the guards technique. In grey we denote the added extra cells. The left pattern combined with the AES diffusion layers is vulnerable to transition leakage, the right provides resistance. The third figure shows the activation of columns of the second pattern combined with `ShiftRows`.

8.1.3 ASCON

ASCON [56] consists of a mode of operation which uses a specific permutation. We focus on the permutation. The round function is depicted in Figure 8.6. The substitution layer is the parallel column-wise application of 64 5-bit S-boxes which are an affine transformation of the χ mapping of Keccak. These S-boxes have linear branch number three. The linear layer consists of five row-wise applications of a linear function Σ. Each Σ function has linear branch number four and uses different rotation values depending on the row to optimize diffusion over several rounds.

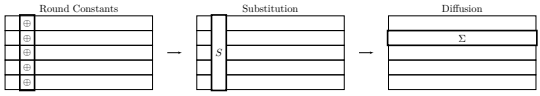


Figure 8.6: The round function of ASCON.

Every Architecture is Secure Consider again an arbitrary glitch-extended probing secure masking of ASCON. We investigate whether we can transform this masking to be transition-extended probing secure. First, while the S-box of ASCON has a nontrivial linear branch number, its sharing might not have a similar property. Denoting the ASCON S-box by S. While S has a correlation zero transition between its first input bit x and first output bit z, the sharing of S can allow for such a transition. Meaning that one can place a nontrivial mask on the sharing of x and on the sharing of z and still find nonzero correlation. In the original work [48], we show that a sharing of the S-box with a nontrivial branch number over the bits exists. Formally, this S-box has a branch number in the following sense, given that $\mathbb{V} = \bigoplus_{i=1}^{5} \mathbb{V}_i$ such that each mask $u \in \mathbb{F}_2^{20}/\mathbb{V}^{\perp}$ can be decomposed in $(u_1, u_2, u_4, u_4, u_5)$ with $u_i \in \mathbb{F}_2^4/(\mathbb{V}_i)^{\perp}$ then

$$\min_{\alpha,\beta \in \mathbb{F}_2^{20}/\mathbb{V}^{\perp}, C^{\tilde{S}}_{\alpha,\beta} \neq 0} \{\mathrm{wt}(\alpha) + \mathrm{wt}(\beta)\} = 3\,.$$

In our work, we also found uniform sharings of the ASCON S-box which did not have the above property.

In case the sharing of the ASCON S-box has a nontrivial branch number, a round-based implementation is automatically secure against any harmful transition leakage. A typical simplified activity pattern is shown in Figure 8.7. Moreover, the same applies for other architectures like a bit-serial implementation. In other words, except for some trivial share-serial approaches, the round function of ASCON requires no additional care to prevent any harmful transition leakage.

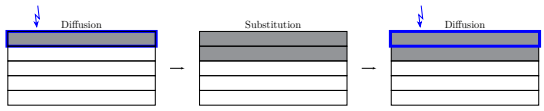


Figure 8.7: A simplified activity pattern in ASCON caused by a transition-extended probe.

8.2 Analysing Coupling Leakage

We consider a leakage effect originating from coupling capacitors between circuit wires, and between circuit wires and ground which are influenced by the switching activity on that wire causing recombinations of the wire values. Effectively when observing leakage from one wire, one can observe leakage from nearby coupled wires. De Cnudde *et al.* [41] has shown that the security of masked hardware implementations can be affected due to coupling effects. These effects are integrated into the probing model following the work by Faust *et al.* [59].

"**Specific model for couplings.** For any set of adjacent wires $\{w_1, ..., w_d\}$, routing recombinations (aka couplings) can be modeled with c-extended probes so that probing one wire w_i allows the adversary to observe c wires adjacent to w_i."

Defending against the above model is not straightforward as two wires carrying different shares of a secret can be coupled. The work by De Cnudde *et al.* [42] discusses three potential solutions to "separate" logic.

- To perform sequential operations instead of parallelism where the implementation processes a non-complete set of shares at each clock cycle. While this provides security this reduces the throughput and avoids making use of the full parallelism feature of hardware.
- Via the use of embedded voltage regulators (VRM) inside the chip, which are already used in commercial smart cards. However, it remains unclear whether electromagnetic signals exhibit potential issues.
- Via the chip having separate Vdd lines to supply functions associated to each share independently. However, it is still an open problem how to supply nonlinear functions which operate on sets of shares and prove the security of the countermeasure.

We assume one of the above countermeasures has been applied to the masking. To that end, we assume the masking is "domain non-complete" meaning that each domain processes only a non-complete set of shares per cycle. An example is shown in Figure 8.8. As a result, a coupling-extended probe (for any c in the above definition) does not yield all the shares of a secret. This non-completeness property alone is sufficient to protect an implementation against coupling-extended probes. However, a more detailed security analysis is needed when combining multiple leakage effects together.

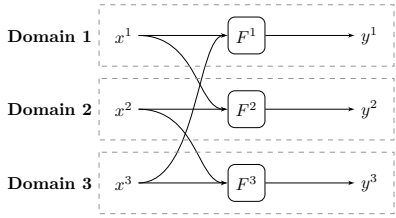


Figure 8.8: Separation into domains of a layer of masked Boolean functions.

8.3 Analysing Glitches, Transitions, and Couplings

We study maskings which are secure against the combined effect of glitches, transition leakage, and coupling leakage. Considering coupling effects, we study the effect when $c = 1$ following the definition above. That means the adversary is capable of observing the probed wire along with one coupled wire. Combining coupling effects with glitches and transitions, using a single probe, the adversary observes all inputs to the probed wire, the previous values which flowed through those resources, and glitches and transition leakage from a coupled operation. We revisit the case studies from Section 8.1.

8.3.1 PRESENT: Serial Architecture

We revisit the masking of PRESENT from Section 8.1.1. More specifically, we consider a glitch-extended probing secure masking in an S-box serial architecture which we showed was secure against the combined effect of glitches and transitions. We adapt the architecture to work share-serial meaning that each cycle one share of one S-box (or quadratic function in the S-box) is calculated. In particular, such an architecture is domain non-complete.

In this case, glitch effects already include the effect of couplings as only one operation per cycle is calculated. When the architecture would calculate two shares of the same S-box consecutively, transitions could leak the input secret of the S-box. Instead, we interleave computation on shares of a secret with the computation of other parts of the state. Thus, the architecture would first calculate the ith share of each S-box before calculating a $(i+1)^{\text{th}}$ share. This secures from harmful transition leakage. The activity patterns are the same as before, given in red in Figure 8.1.

8.3.2 AES: Serial Architecture

We consider a glitch-extended probing secure masking in a serial architecture where only one S-box and `MixColumns` is implemented. This is a popular architecture for maskings, examples can be found in the works by De Meyer *et al.* [47, Figure 6] and Gross *et al.* [62, Figure 5]. In this case, we consider that a domain separation through different Vdd lines or via embedded voltage regulators has been implemented such that the design is domain non-complete.

Even though there is only one masked S-box on the implementation, it typically consists of multiple register stages. Thus, the S-box can compute on several bytes at once in a pipelined manner. Recall that we assume that $c = 1$, thus a probe on an S-box views the computation of another coupled S-box. Due to transition leakage, the adversary can view the two S-boxes over two consecutive cycles. Due to the `MixColumns` having branch number five, this leakage can never reveal a secret. However, there is still leakage from the calculation of the `MixColumns` as noted in Section 8.1.2. Finally, there can be coupling leakage between the `MixColumns` and an S-box (from a different column). However, these cases do not add extra harmful activity patterns. Thus, as explained in Section 8.1.2, we only require adding one cell of extra randomness to secure against the combined effect of glitches, transitions, and couplings.

8.3.3 ASCON: Round-Based Architecture

Finally, we consider a glitch-extended probing secure masking with a round-based architecture for ASCON. Again, we consider the case where $c = 1$ meaning that a probe can observe an additional operation in that cycle and we consider domain non-complete maskings where there is some countermeasure ensuring coupling leakage alone does not reveal all shares of a secret.

Due to the combined effect of transitions and couplings, a probed S-box gives information on two S-boxes over two consecutive rounds. Previously, we argued

ASCON was secure when probing an S-box due to the nontrivial branch number of the linear layer. This argument no longer holds as the linear layer only has branch number four. However, the case remains secure as the layer does not allow for transitions of two active input bits to two active output bits. This can be verified on sight from the equations of the Σ function:

$$\Sigma_{\alpha,\beta}(x) = x \oplus (x \ggg \alpha) \oplus (x \ggg \beta)\,,$$

with $\ggg$ the right circular shift and $\alpha, \beta \in \mathbb{N}$ constants specific for ASCON.

When the adversary probes the linear layer, it can potentially activate two input and two output bits of a shared S-box. The diffusion of the S-box is not sufficient to prevent harmful leakage. However, we observe that the probe returns bits from the same share, e.g. both from the first share of the input and first share of the output. Thus, one can protect the implementation when the shared S-box $\bar{S}$ has the following property. Given an s-sharing $\bar{S}$ and $\mathbb{V} = \bigoplus_{i=1}^{s} \mathbb{V}^i$ such that each mask $u \in \mathbb{F}_2^{5s}/\mathbb{V}^\perp$ can be decomposed in $(u^1, ..., u^s)$ with $u^i \in \mathbb{F}_2^5/(\mathbb{V}^i)^\perp$ then for $i \in \{1, ..., s\}$

$$\forall \alpha, \beta \in \mathbb{F}_2^{5s}/\mathbb{V}^\perp :\ C^{\bar{S}}_{\alpha,\beta} = 0 \text{ when } \alpha = (0, ..., \alpha^i, ..., 0), \beta = (0, ..., \beta^i, ..., 0)\,.$$

A similar property was explored in Section 8.1.2 on diffusion between shares. Since the S-box is quadratic, one can easily find non-complete sharings of the entire S-box. This non-completeness can be used to argue that the same input and output share of the S-box do not reveal any secret information.

8.4 Conclusion

The above case studies show that often the diffusion layers of symmetric primitives already provide quite good protection against transition leakage. At most we need to use only a few bits of extra randomness to protect the entire masking against this more advanced leakage effect. This should be compared to gate-level countermeasures like the work by Cassiers and Standaert [28, Table 2] requiring an extra several thousands of bits of randomness to secure a glitch-resistant PRESENT against transition leakage. We have also shown that the model easily adapts to other advanced leakage effects such as coupling leakage. In short, because the analysis method works primitive-wide, it becomes easy to investigate more advanced adversaries.

Observe that we never used the nonlinearity of the sharing of the S-box, meaning we only relied our security analysis on the diffusion of the masking. Interesting future work would be to create a tool which can automatically verify whether

a netlist is secure against transition leakage. This tool could work by using a circuit model comprised of XOR, AND gates, and branches and using their linear cryptanalytic diffusion properties to propagate back to the previous value in that resource.

Chapter 9

Fault Injection and Countermeasures

In the previous chapters, we explored the threat of side-channel attacks and masking as their countermeasure. In this chapter, we study a different attack vector against embedded devices, namely fault attacks. These attacks have the goal to make errors in the computation of the device such that it outputs the wrong value. In turn, those faulted outputs can then be used by the adversary to retrieve secret data. We investigate how to protect against fault attacks and the combination of fault attacks and side-channel attacks. More specifically, we investigate adversary and security models.

Sections 9.5, 9.6.2, 9.6.3, and 9.6.4 are based on my published works, the other sections are based on previous work. The writing of this chapter is based on two works [49, 50] published at CARDIS and CT-RSA. These works are made by myself and supervised by Svetla Nikova.

9.1 Motivation

The motivation for this section differs from that of the previous ones. Here, we do not aim to reduce as much randomness as possible. We revert to the easier models where we assume that ideal and efficient random bit generators exist.

The field of *active security*, or fault security, is largely under-researched. The field of *combined security*, where the adversary uses both fault attacks as side-

channel attacks, even more so. As a result, no clear adversary or security models are known. The first step in this research topic should not yet be to increase efficiency or question leak-free or fault-free components such as the generation of randomness, but instead to propose preliminary adversary and security models. The aim is to find a balance between theory and practice. Meaning, to find a model which is sufficiently close to practice, but still allows for efficient and easy-to-verify countermeasures. As such, the first step of the research is to generalise the notion of strong non-interference as seen in Section 2.6.2.

9.2 Quick Overview of Fault Attacks

In this section, we give an overview on fault attacks. We introduce both the physical methods and how they can be used to attack the cryptographic assumptions of algorithms.

9.2.1 Fault Injection Mechanisms

We outline three popular methods to inject a fault in the computation. We refer the reader to the preliminaries by Richter-Brockmann *et al.* [90] for more information on these fault mechanisms.

Clock Glitches With clock glitches, the attacker has some access to the clock present on the device. By tampering with this clock, the attacker can inject faults by disturbing the computation on the critical path of that cycle. This allows for a cheap and controlled way to inject faults. However, there are also cheap methods to protect the clock in ASICs.

In an adversarial view, clock glitches allow for precise fault attacks which are easily repeated each clock cycle. However, the position of the fault is not in the control of the adversary as it is determined by the critical path of the implementation.

Electromagnetic Pulses With such pulses, the attacker creates a local electromagnetic event in the implementation. The sudden change of electromagnetic radiation causes current to flow through the affected wires, in turn causing voltage swings.

In an adversarial view, the attack affects a larger area on the device and is often not precise. The attack also mainly targets memory elements instead of the combinatorial logic.

Optical Fault Injections Using lasers, the adversary can inject faults on a transistor level in implementations. Lasers cause ionising processes in the PN junction of MOS transistors. These attacks can be used to alter the state of a gate.

In an adversarial view, optical faults can be extremely precise but they are difficult to use when wanting to inject multiple faults at once since the laser is typically focused on only one resource in the platform. However, a laser setup does make it possible to fault the same resource in the platform multiple times. The fault also has permanent effects making it difficult to experiment with this fault source in practice as it easily destroys the platform.

9.2.2 Breaking Cryptographic Assumptions

Previously, we discussed how attackers can inject faults. In this part, we discuss how faults can be used to break the cryptographic assumptions of symmetric-key primitives. One of the first works on this area dates back from Biham and Shamir [13] in 1997. They introduced Differential Fault Analysis (DFA), a key-recovering cryptanalytic technique applicable to any symmetric-key cryptosystem. Their work demonstrated the attack on DES. The power of DFA lies in the fact that typically only few successful faults are needed, for example the attack on the AES by Moradi *et al.* [78] only requires six faulty ciphertexts.

Fault attacks are still effective even if the symmetric primitive is masked. In particular, the method of *Statistical Ineffective Fault Analysis* (SIFA) proposed by Dobraunig *et al.* [55] in 2018 has been particularly effective as it not only threatens the security of masking methods but also many previously existing countermeasures against fault attacks. The attack essentially injects a fault and verifies whether either the output is incorrect or an existing fault detection method detected the fault. In case the fault was not detected and the ciphertext came out correct, the adversary can learn secret information. The core idea is that, due to faults, the adversary can assume that the fault did not change the state of the computation and, as a result, deduce the value of this state thus breaking the privacy of the computation.

9.3 Linear Codes and Redundancy

To defend an algorithm against fault attacks, the core idea is to utilise redundancy to enable detection of the injected faults. This redundancy is found in encoding intermediate variables using error detecting codes. A popular encoding method is to duplicate intermediate variables, such that, by checking whether all duplicates are equal, an algorithm can detect injected faults. More specifically, a variable x is encoded to $(x^1, ..., x^k)$ such that $x = x^1 = ... = x^k$. This last equality check is the method to verify whether there are errors present on the computation.

Essentially, the idea is that the adversary can only fault a single part of the computation at a time and that the probability to inject the same fault in multiple parts is negligibly low. As a result, such a duplication method works in practice.

Other encoding techniques have been studied in the literature. For example, there is an active secure version of polynomial masking [29]. Other techniques using a multiplicative message authentication code also exist, see for example the works by Reparaz *et al.* [89] and De Meyer *et al.* [45].

9.4 The Adversary and Security Model

In this section, we cover the current standard adversary model for combined adversaries. We also consider some extensions to the fault adversary.

9.4.1 The Combined Adversary and Security Models

We immediately consider the combined adversary which interacts with a circuit by both observing parts of its computation and injecting errors into it.

Consider passive, active, or combined adversaries as those which interact with a circuit by placing probes, faults, or both respectively. The adversary's strength is determined by which and how many wires in the circuit it can probe or fault. For example, the capability of an active attacker can vary from faulting a single wire (e.g., using a laser) or targeting several wires at once; attackers can trigger a fault once or multiple times (even in a single cycle). The fault may effect the value on the wire in one cycle or during multiple cycles or even for the total protocol execution.

After the adversary has chosen which wires to probe and fault, the circuit reacts by setting or toggling the values on the faulted wires (as specified by the adversary) and returning the values on the probed wires. The state of the abort signal (true or false) is returned as well. Formally, for $t = 0, ..., \ell$, a (d, k)-threshold-probe-and-faulting adversary on $\mathbb{F}_{2^m}^{\ell}$ is a machine $\mathcal{A}$ that plays the following game against an oracle $\mathcal{O}$:

1. $\mathcal{A}$ specifies a set $\mathcal{I} = \{i_1, ..., i_{|\mathcal{I}|}\}$ of cardinality at most d and a set of tuples $\mathcal{J} = \{(j_1, h_1), ..., (j_{|\mathcal{J}|}, h_{|\mathcal{J}|})\}$ of cardinality at most k,
2. $\mathcal{O}$ computes an arbitrary sequence $(x_1, ..., x_\ell) \in \mathbb{F}_{2^m}^{\ell}$ with $(x_{j_1}, \ldots, x_{j_{|\mathcal{J}|}})$ faulted (bit-flip or stuck-at) according to $(h_{j_1}, \ldots, h_{j_{|\mathcal{I}|}})$.
3. $\mathcal{A}$ receives $(x_{i_1}, \ldots, x_{i_{|\mathcal{I}|}})$ with the abort state $\perp$.

We also cover the security model following the work by Ishai *et al.* [68]. Security consists of both a privacy and a correctness aspect. The number of probe and faults an adversary needs to place to break the scheme determines the order of security.

Definition 9.1 (Order of combined security [50])**.** A circuit is (d, k)-*order combined secure* if the following holds against a (d, k)-threshold-probe-and-faulting adversary.

1. Privacy: The probed d-tuple with the state of the abort can be simulated from scratch.
2. Correctness: The circuit either aborts $\perp$ or gives a correct output.

9.5 The NINA Security Model

In this section, we introduce our NINA security model. Essentially, this model is the active and combined secure counterpart of the non-interference model from Section 2.6.2.

9.5.1 The Simulation Model

We aim to protect against passive, active or combined adversaries as those who interact with a circuit by placing probes, faults, or both respectively. The circuit responds to this adversary by setting or toggling the values on the faulted wires

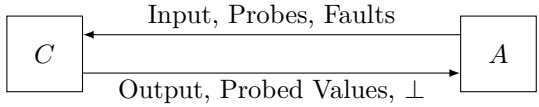


Figure 9.1: Interaction between a circuit C and an adversary A.

and returning the values on the probed wires. The state of the abort signal (true or false) is returned as well. This is shown in Figure 9.1.

In order to make simulation based proofs for the secrecy of shared variables in circuits, we define simulatability similar to the definition given earlier in Definition 2.4. However, we additionally consider that up to k wires in that circuit have been faulted and that the circuit can abort. Here the adversary (distinguisher) is either interacting with the actual circuit or with a simulator. This simulator is given only a part of the input and does not know the secrets of the circuit. The distinguisher's goal is to determine whether it is interacting with the simulator or with the actual circuit. A failure to do so implies that the adversary can know at most the shares given to the simulator and as a result only some inputs of the circuit.

Definition 9.2 (Simulatability)**.** Let $P = \{p_1, ..., p_d\}$ be a set of d probes of a circuit C with m inputs where each input is divided into n shares. Let the set of q shares of each input given to the simulator be denoted by $I = \{(i_1, j_1), ..., (i_m, j_q)\} \subset \{1, ..., m\} \times \{1, ..., n\}$. Let $F = \{(f_1, e_1), ..., (f_k, e_k)\}$ be a set of k injected faults e_i (either set or add) on the wire f_i in C. Denote $C_{P,F}$ as the circuit C with probed wires as per P and injected faults as per F. Finally, let $\perp \in \{0, 1\}$ denote the state of the abort signal in the circuit.

We define the simulator S and distinguisher D as the following probabilistic functions.

$$S : \mathbb{F}^q \times \mathbb{F}^m \times \mathbb{F}^k \to \mathbb{F}^d \times \{0, 1\}$$

$$D : \mathbb{F}^d \times \{0, 1\} \times \mathbb{F}^k \times \mathbb{F}^{nm} \to \{0, 1\}$$

We say that the set of probes P and the state of the abort signal $\perp$ of the faulted circuit C_F can be simulated with the set of values on the input wires I if there exists a simulator S, such that for any distinguisher D and any inputs $a^{*,*}$, we have that

$$\left| \Pr[D(C_{P,F}(a^{*,*}), F, a^{*,*}) = 1] - \Pr[D(S(I, F), F, a^{*,*}) = 1] \right|$$

is negligible in the passive threshold of the sharing scheme, where the probability is taken over the random coins in C, S and D.

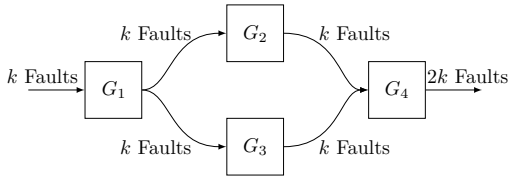


Figure 9.2: An example of the propagation of faults over several k-NA circuits for which a stronger composability notion is needed.

We note that for composable security, as we will see later on, we require that the probability for the distinguisher to view the difference between the circuit and the simulator is negligible and we should take care composing circuits when it is not.

9.5.2 Non-Accumulation

We defined the order of active security as the maximal number of faulty wires such that the circuit still returns a correct output. We now make this into a composable notion, thus we look at the effect of a fault in a circuit which is part of a larger whole. Ideally an injected fault in the circuit is not propagated, *i.e.* the fault does not affect the output of that circuit. However, the adversary can always fault its output directly, meaning that we can never guarantee that all outputs of a faulted circuit are correct. Instead, we are interested in circuits which do not accumulate faults. In other words, we need a fault on a single input or intermediate wire to affect only a single output of the circuit. We relax this requirement by allowing countermeasures to abort the computation (e.g., by using error detecting methods). We thus find composable active security notions which are similar in nature to the definitions of NI and SNI discussed earlier. Our first notion is Non-Accumulation (NA).

Definition 9.3 (k-Non-Accumulative (k-NA))**.** A circuit G is k-NA if for any set of $k' \leq k$ errors, the circuit either aborts or gives an output with at most k' errors.

For a circuit which is k-NA, k faults on its intermediate variables result in the circuit giving an output with at most k faults. When composing circuits, a stronger notion of non-accumulation is needed to guarantee the security of the composition. For example, consider the case given in Figure 9.2 where each circuit G_i is k-NA. If an adversary injects k faults in the input of G_1, the circuit will give an output with at most k faulty shares. These faults propagate to the inputs of G_2 and G_3 which, because both circuits are k-NA, results in a

worst case scenario where G_4 gets an input with a total of $2k$ faulty shares. The end result is a sharing with $2k$ faulty shares even though only k faults were injected. To avoid such an accumulation of faults, one needs circuits which are capable of erasing the errors from their input. The following definition of Strong Non-Accumulation (SNA) is sufficient to arbitrarily compose circuits and be assured of their active security.

Definition 9.4 (k-Strong Non-Accumulative (k-SNA))**.** A circuit G is k-SNA if for any set of k_1 errors on each input and every set of k_2 errors on the intermediate variables, with $k_1 + k_2 \leq k$, the circuit either aborts or gives an output with at most k_2 errors.

When the non-accumulation notions are combined with a sharing scheme with a high enough active threshold, the composable notions provide active security.

9.5.3 NINA

We now look at composable security notions considering circuits which are both probed and faulted. First, we need to guarantee the correctness of the output of each circuit. To capture the effect of faults in compositions of circuits, we use an argument similar to the one on active security. Thus,we need that an injected fault in a circuit propagates to at most one output share. However, the adversary can now place probes and thus learn part of the computation made in the circuits. As a result, the combined security notion needs to capture the probability of an adversary breaking the correctness of a circuit given several faults and probes. In this work, we only propose countermeasures where the correctness can not be broken, to give an example of a countermeasure for which this probability is non-trivial we refer the reader to the countermeasure by Reparaz *et al.* [89]. Apart from guaranteeing the correctness of a circuit, we also guarantee its sensitive variable privacy for which we use simulation based arguments similar to non-interference. As mentioned by Clavier et al. [31], fault injections can act as a probing tool (think of an adversary faulting away the randomness in a circuit). Thus, we treat faults as probes giving extra shares to the simulator per injected fault. Additionally, to give the designer the freedom to make countermeasures more efficient we consider security with abort. To capture the effect of the abort signal potentially revealing secrets in the circuit, we require the simulator to reproduce this signal given the injected errors and some input shares. As a result, we design a composable security notion of order (d, k) such that the circuit is (d', k')-order combined security for all sets of $d' + k' \leq d$ probes and $k' \leq k$ faults. The notion is called NINA derived from the concatenation of the names Non-Interference (NI) and Non-Accumulation (NA).

Definition 9.5 ((d,k)-NINA)**.** A circuit G is (d,k)-NINA if for any set of $k' \leq k$ errors and any set of d' probes, such that $d' + k' \leq d$, the following holds.

1. Privacy: The probes and the abort signal can be simulated with $d' + k'$ shares of each input and the injected errors.
2. Correctness: The circuit either aborts or gives an output with at most k' errors.

The NINA notion, combined with a sharing scheme having a sufficient passive and active threshold, implies the notion of combined security of Definition 9.1. This follows from the simulation based security stating that the adversary can learn up to a threshold number of the circuit's inputs which, if lower than the passive threshold of the sharing scheme, gives no information on the circuit's secrets. Similarly, since the adversary can only fault up to a threshold number of outputs, a decoding circuit can detect or correct those errors given that the sharing scheme has enough redundancy in it.

Theorem 9.1. *A (d,k)-NINA circuit G with input encoding $\mathcal{I}$ and output decoding $\mathcal{O}$ using a secret sharing scheme with passive threshold at least d and active threshold at least k is (d',k')-order combined secure for any $d' + k' \leq d$ and $k' \leq k$.*

Proof. We recall that an input encoding function $\mathcal{I}$ is a probabilistic circuit which shares the input using uniform randomness and is called before the circuit G. The output decoding function $\mathcal{O}$ gives back the decoding of its input shares or an abort signal $\perp$ and is called on the output of G. We take an arbitrary d', k' set of probes and faults such that $d' + k' \leq d$ and $k' \leq k$ and prove the correctness and privacy of G with $\mathcal{I}$ and $\mathcal{O}$ following Definition 9.5.

First, we prove that the adversary does not violate the correctness of G with $\mathcal{I}$ and $\mathcal{O}$ given the knowledge of any d' intermediate values and any set of k' faults in the intermediate values. From Definition 9.5 of NINA we know that G gives an output with less than k' faulty shares or an abort signal. Since $k' \leq k$, we know from our secret sharing scheme that the output decoder $\mathcal{O}$ either corrects its input and gives a correct output or detects an error and aborts the computation.

Second, we prove the privacy of G with $\mathcal{I}$ and $\mathcal{O}$. Since we defined simulatability (Definition 9.2) with computational security, we consider d large enough for the probability of simulation failure for G to be negligible. From Definition 9.5 of NINA we know that the adversary learns at most $d' + k'$ input shares of G from the injected probes and faults. Since $d' + k' \leq d$, the passive security

threshold of the secret sharing scheme is not reached and hence no information of G's secret inputs are revealed due to the properties of $\mathcal{I}$. Additionally, the adversary is given the state of the abort signal of the circuit. From the definition of NINA we know that given those $d' + k'$ input shares, we can simulate the abort signal, i.e., we can decide whether or not the abort signal was flagged. As we simulated the abort signal and the faults on the output shares, we can also simulate whether $\mathcal{O}$ gives back a correct output or an abort. As a result, the privacy of G is proven. □

As a result, if we prove a circuit is NINA, we know it is combined secure. However, just as with non-interference, the NINA notion is not sufficient for composability. To this end we introduce *Strong NINA* (SNINA).

Definition 9.6 ((d, k)-SNINA)**.** A circuit G is (d, k)-SNINA if for any set of k_1 errors on each input and k_2 intermediate errors, any set of d_1 intermediate probes, any set of d_2 probes on the output, such that $d_1 + d_2 + k_1 + k_2 \leq d$ and $k_1 + k_2 \leq k$, the following holds.

1. Privacy: The probes and the abort signal can be simulated with $d_1+k_1+k_2$ shares of each input and the injected errors.
2. Correctness: The circuit either aborts or gives an output with at most k_2 errors.

The notion of SNINA is sufficient for composability. In other words the composition of two SNINA circuits is again SNINA.

Theorem 9.2. *The composition of two (d, k)-SNINA circuits is (d, k)-SNINA.*

Proof. Consider two circuits G_1 and G_2 and their composition G_3. We prove that G_3 is (d, k)-SNINA given that both G_1 and G_2 are (d, k)-SNINA. We assume that G_1 and G_2 share at least one wire otherwise the proof is evident. Without loss of generality we assume the output of G_2 is not an input to G_1 (thus G_1 is evaluated before or at the same time as G_2). Since we defined simulatability (Definition 9.2) with computational security, we consider the passive threshold of the sharing scheme to be large enough for the probability of simulation failure for G_1 and G_2 to be negligible.

Take an arbitrary set of k_1 errors on each input of G_3, arbitrary sets of k_2 intermediate errors and d_1 probes on G_3, and an arbitrary set of d_2 probes on the output of G_3, such that $d_1 + d_2 + k_1 + k_2 \leq d$ and $k_1 + k_2 \leq k$. Because G_3 is the composition of G_1 and G_2, we divide the d_1 probes as d_3 probes on G_1 and d_4 probes on G_2 such that $d_3 + d_4 = d_1$. Similarly, we have k_3 faults on G_1 and k_4 faults on G_2 such that $k_3 + k_4 = k_2$.

We first prove the correctness of G_3, since G_1 is instantiated with k_1 errors on each input and k_3 intermediate errors, such that $k_1 + k_3 \leq k$, and since G_1 is (d, k)-SNINA it either aborts or its output has at most $k_3 \leq k_2$ errors on it. Now G_2 is instantiated with either k_1 or k_3 faults on each input and has k_4 intermediate faults, since G_2 is (d, k)-SNINA and $k_1 + k_4 \leq k$, $k_3 + k_4 \leq k$, it follows that G_2 either aborts or its output contains at most $k_4 \leq k_2$ errors. Since the combined outputs of G_1 and G_2 have less than k_2 faults on them, we have proven the correctness of G_3.

Second, we prove the privacy of G_3. First if G_1 aborts, then we can simulate this abort signal with $d_3 + k_1 + k_3$ of G_1's input shares and the injected errors. Recall that G_2 is instantiated with either k_1 or k_3 faults on each input and there are k_4 faults on its intermediate variables. Similarly, there are d_2 probes on G_2's output and d_4 intermediate probes. Since G_2 is (d, k)-SNINA, the adversary learns at most $d_4 + k_4$ input shares of G_2. Additionally, the adversary learns $\max(k1, k3)$ shares due to the faults on G_2's inputs. Since both G_1 and G_2 are (d, k)-SNINA and less than k faults are injected, we know that each fault causes the adversary to learn at most one share of each input per fault. From this we learn that the k_3 injected faults in G_1 cause the adversary to learn k_3 input shares of G_1 and since at most k_3 faults propagate to its output, the adversary learns at most k_3 input shares of G_2. We also see that the k_1 input faults on G_3 give the adversary knowledge of at most k_1 shares of G_1 and of G_2, however these do not propagate from G_1 to G_2. Finally, since the output of G_1 is either the output of G_3 or the input to G_2 and since $d_3 + d_4 + d_2 + k_1 + k_3 + k_4 \leq d$, we can simulate the probed values and the abort signals of G_1 and G_2 with at most $d_3 + d_4 + k_1 + k_3 + k_4 \leq d_1 + k_1 + k_2$ shares of each input. This proves the privacy of G_3 and thus that G_3 is (d, k)-SNINA. □

The above theorem together with Theorem 9.1 implies that the notion of SNINA is a sufficient condition to achieve composable combined security.

9.5.4 A Combined Secure Multiplier

We provide a multiplier which can be proven secure in the NINA model described above. For the proposed countermeasure, we make use of a duplicated Boolean masking approach which shares a secret x as a vector

$$(x^{1,1}, ..., x^{1,k+1}, x^{2,1}, ..., x^{d+1,k+1}),$$

such that $\sum_{i=1}^{d+1} x^{i,\ell} = x$ for all $\ell \in [k+1]$ and $x^{i,1} = ... = x^{i,k+1}$ for all $i \in [d+1]$. This method has a passive threshold d meaning that no d shares

give information on the secret x and an active threshold k meaning that any faults on at most k shares could be detected in the share vector.

The pseudo-code of the multiplication circuit is given in Algorithm 9.1. The circuit starts by multiplying two independent share vectors of a and b to create all cross products of the form $a^i b^j$. These cross products are then re-masked by adding unique randomness $r^{i,j}$. Since we add the same randomness over all duplicated cross products ($u^{i,j,\ell}$ for $\ell \in [k+1]$) all these cross products should equal each other if no fault was injected. As a result, we can verify them for potential errors. To detect errors in the cross products it is sufficient to compare a share to all its duplicated versions, in symbols:

$$\forall i, j \in [d+1], \forall \ell \in [k+1] : \; u^{i,j,1} = u^{i,j,\ell} \, .$$

Since we are working over bits, this translates to aborting the computation in case one of the $u^{i,j,1} + u^{i,j,\ell}$ is equal to 1. This abort operation is considered as a command causing all variables in the implementation to read $\perp$. In case no error is detected, the circuit sums up all the cross products for different indices j and returns a duplicated Boolean sharing of ab. From the proof of security we see that there is a statistical ineffective fault attack which breaks the privacy of the algorithm. This attack works as follows, the adversary adds a non-zero fault to one of the $a^{i,\ell}$ shares (similarly $b^{i,\ell}$ shares). In case the operation does not abort, the adversary learns that all $b^{i,\ell} = 0$ (similarly all $a^{i,\ell} = 0$), which means the adversary learns an input secret and breaks the privacy of the circuit. The probability for this attack to succeed is equal to $1/|\mathbb{F}_2|^{d+1}$. Due to the attack aborting the computation when it fails, this attack does not threaten the composability of the circuit.

9.5.5 Combined Security of the Multiplication

Now we prove the SNINA security of Algorithm 9.1. Intuitively, the proof is a combination of the SNI and SNA security where the simulator now needs to simulate the abort signal. This simulator is given a share of each input per probe and fault injected in the gadget.

We start by showing the multiplier is strong non-accumulative.

Theorem 9.3. *Algorithm 9.1 is k-SNA.*

Proof. Take an arbitrary k_2 faults on the intermediate variables and k_1 faults on each input such that $k_1 + k_2 \leq k$. We show that Algorithm 9.1 either aborts or gives back a result with at most k_2 errors. We classify the internal variables in the following groups.

Algorithm 9.1: Multiplying duplicated Boolean shared values

Input: Independent shares of a and b, and uniform random $r^{i,j}$
Output: Shares of ab or $\perp$

for $\ell \leftarrow 1$ **to** $k+1$ **do**
  for $i \leftarrow 1$ **to** $d+1$ **do**
    $u^{i,i,\ell} \leftarrow a^{i,\ell} b^{i,\ell}$;
    for $j \leftarrow i+1$ **to** $d+1$ **do**
      $u^{i,j,\ell} \leftarrow a^{i,\ell} b^{j,\ell} + r^{i,j}$;
      $u^{j,i,\ell} \leftarrow a^{j,\ell} b^{i,\ell} + r^{i,j}$;
    end
  end
end
for $\ell \leftarrow 2$ **to** $k+1$ **do**
  for $i \leftarrow 1$ **to** $d+1$ **do**
    for $j \leftarrow 1$ **to** $d+1$ **do**
      $t^{i,j,\ell} \leftarrow u^{i,j,1} + u^{i,j,\ell}$;
      if $t^{i,j,\ell} = 1$ **then return** $\perp$;
    end
  end
end
for $\ell \leftarrow 1$ **to** $k+1$ **do**
  for $i \leftarrow 1$ **to** $d+1$ **do**
    $c^{i,\ell} \leftarrow \sum_{j=1}^{d+1} u^{i,j,\ell}$;
  end
end

1. $a^{i,\ell}, b^{i,\ell}$
2. $r^{i,j}$
3. $u^{i,j,\ell}$
4. $t^{i,j,\ell}$
5. $c^{i,\ell}$

We go over all possible faults and determine whether the algorithm aborts or which outputs $c^{i,\ell}$ are affected. Recall that Algorithm 9.1 has a total of $(d+1)^2 k$ error checks by verifying that $u^{i,j,1} = u^{i,j,\ell}$ for all $i, j \in [d+1]$ and $\ell \in [k+1]$. We distinguish the following cases.

- The adversary used no faults then the algorithm gives back a correct result.
- The adversary injected a fault in group (1). For ease we consider that a^i was faulted. We go over the following sub-cases.
  - No cross products $u^{i,j,\ell}$ are faulty. In this case all $b^{j,\ell}$ for $j \in [d+1]$ were equal to zero. As a result, the faults on the a shares are removed, the algorithm does not abort and gives back a correct output.
  - At least one $u^{i,j,\ell}$ is faulty. In this case there are k error checks performed on the faulty $u^{i,j,\ell}$. Since the adversary can only inject $k-1$ other faults, it can not bypass all checks and the algorithm aborts.
- The adversary injected a fault in group (2). This fault affects all $u^{i,j,\ell}$ and $u^{j,i,\ell}$ for $\ell \in [k+1]$. As a result, this fault goes undetected. However, it does not affect the correctness of the algorithm as the output is still a duplicated Boolean sharing of ab.
- The adversary injected a fault in group (3). This fault would be detected by the k error checks on $u^{i,j,\ell}$ for $\ell \in [k+1]$. As the adversary can only fault $k-1$ other variables, it can not bypass these checks and the error will be detected.
- The adversary injected a fault in group (4). As this value is only used to check whether an error is present on the cross products, it does not influence the output of the algorithm.
- The adversary injected a fault in group (5). In this case $c^{i,\ell}$ is faulty.

As the only faults which affect the output shares are those in group (5) and since the adversary can only inject k_2 intermediate errors, Algorithm 9.1 gives an output with at most k_2 erroneous output shares or it aborts. Thus we have proven that Algorithm 9.1 is k-SNA. □

Given the above property of strong non-accumulation, we can move over to the combined security of the multiplier. We note that the following proof shows there is an attack which has a probability of violating the privacy of the gadget. This probability can be made negligible in d.

Theorem 9.4. *Algorithm 9.1 is* (d,k)*-SNINA.*

Proof. The proof is split in two parts. In the first part we prove that a probing and faulting adversary can not get an output with more than the injected

number of errors of erroneous shares. In the second part we prove that this adversary also can not break the privacy of the scheme.

We take arbitrary inputs and injected randomness. Take arbitrary k_1 errors on each input and k_2 intermediate errors such that $k_1 + k_2 \leq k$. We take d_1 arbitrary probes on the intermediate variables and d_2 probes on the output such that $d_1 + d_2 + k_1 + k_2 \leq d$.

First, we prove that given the faults and probes, the algorithm does not give an output with more than k_2 erroneous outputs. From Theorem 9.3 we know that for any set of k_1 errors on each input and k_2 errors on the intermediate variables, the gadget does not give an output with more than k_2 erroneous shares. Thus, the information given by the probes can not help the adversary to create an incorrect output.

Second, we prove the privacy of the scheme. We classify the internal variables in the following groups.

1. $a^{i,\ell}, b^{i,\ell}$
2. $r^{i,j}$
3. $u^{i,j,\ell}$
4. $t^{i,j,\ell}$
5. $c^{i,\ell}$

We define two sets of indices I and J such that $|I| \leq d_1+k_1+k_2$, $|J| \leq d_1+k_1+k_2$ and such that the probed values and the abort signal can be simulated given only the knowledge of $\{a^{i,1}\}_{i\in I}$ and $\{b^{i,1}\}_{i\in J}$. The sets are constructed as follows.

- For every probe or fault as in group (1) add the corresponding i to either I or J.
- For every probe or fault as in groups $(2), (3), (4)$ add i to I and j to J.
- For every probe or fault as in group (5) add i to I and for every probed $u^{i,j,\ell}$, $r^{i,j}$ or $t^{i,j,\ell}$ add j to J.

Since the adversary is allowed to make at most d_1 internal probes, k_1 faults on each input and k_2 intermediate faults, we have that $|I| \leq d_1 + k_1 + k_2$ and $|J| \leq d_1 + k_1 + k_2$. We now show how the simulator behaves for the internal observed values.

- For each observation as in group (1), since the probed input and the injected errors are given to the simulator, it can perfectly simulate this observation.
- For each observation as in group (2), we simulate $r^{i,j}$ as a uniform random variable.
- For each observation as in group (3), the simulator has access to $a^{i,1}$ and $b^{j,1}$ (a priori $a^{i,1} = a^{i,\ell}$ and $b^{j,1} = b^{j,\ell}$). In case $r^{i,j}$ was not directly probed it is simulated as a uniform random variable, otherwise it was already simulated. In case there were errors injected in the $a^{i,\ell}$, $b^{j,\ell}$, $r^{i,j}$ or $u^{i,j,\ell}$, the simulator is given these errors. As a result, the simulator can perfectly simulate the $u^{i,j,\ell}$.
- For each observation as in group (4). By definition $i \in I$ and $j \in J$. Since the simulator is given the injected faults, it knows the $a^{i,1}, b^{j,1}$ and $a^{i,\ell}, b^{j,\ell}$ used in the $u^{i,j,1}$ and $u^{i,j,\ell}$. Thus the simulation follows as in the previous step.
- For each observation as in group (5), by definition $i \in I, J$. In case the simulator knows the algorithm will abort, it simulates the value $c^{i,j,\ell}$ as $\perp$. Otherwise, we assign a random value to every term $u^{i,j,\ell}$ for $j \in [d+1]$. Then if one of the terms $u^{i,j,\ell}, u^{j,i,\ell}, t^{i,j,\ell}, t^{j,i,\ell}$ or $r^{i,j}$ was already probed or faulted, since by definition $i \in I$ and $j \in J$ the simulator knows $a^{i,\ell}$ and $b^{j,\ell}$ and can simulate $u^{i,j,\ell}$. Otherwise $r^{i,j}$ is not viewed or faulted by any other probe or fault and thus is perfectly simulated as a uniform random value.

We now simulate the outputs $c^{i,\ell}$ using no additional information from the inputs. Recall that faults are treated as probes. We have to take into account the following cases.

- In case the simulator knows the abort signal will be flagged, it simulates all $c^{i,\ell}$ as $\perp$.
- If the attacker has already observed some of the internal values, then the partial sums $u^{i,j,\ell}$ or $a^{i,\ell}b^{i,\ell}$ previously probed are already simulated. As for the remaining terms, we note that by definition of the scheme there always exists one random bit $r^{i,j}$ which does not appear in the computation of any other observed element. Therefore the simulator can assign a random and independent value to $c^{i,\ell}$ unless the $c^{i,\ell}$ was directly faulted then the simulator simulates it as the injected fault.
- If the attacker has only observed output shares, then we point out that, following Algorithm 9.1, each of them is composed by d random bits $r^{i,j}$

and at most one of them can enter in the computation of each other output variable $c^{j,\ell}$. Since the adversary may have previously probed at most $d-1$ of them, there exist one random bit $r^{i,j}$ which does not appear in the computation of any other observed value. Thus the simulator can assign a random and independent element to $c^{i,\ell}$ or as an injected fault, completing the proof.

We go through the possible faults and show we can simulate the abort signal. We note that since we treat faults as probes and since the simulator is given the location and value of the errors, it knows if the injected faults were trivial or not, i.e., if the injected faults changed the original value. This also holds for when there are multiple faults on the same variable.

- For each fault as in group (1). In case there was an injected error which differed from the original value, the simulator aborts the algorithm. For brevity, we assume $a^{i,\ell}$ was faulted. From Theorem 9.3 we know that the real algorithm aborts unless all $b^{j,\ell} = 0$ for $j \in [d+1]$ in which case the simulator fails to simulate the real algorithm. However, since the input shares were randomised the probability this event occurs is $1/|\mathbb{F}_2|^{d+1}$. In case more than one input share was faulty, the probability of failing the simulation is smaller than $1/|\mathbb{F}_2|^{d+1}$.
- For each fault as in group (2), the simulator does not abort. From Theorem 9.3 we know that in case there is an error on the $r^{i,j}$, the algorithm never aborts thus this simulation is perfect.
- For each fault as in group (3), the simulator goes over all errors on $a^{i,\ell}$, $b^{j,\ell}$, $r^{i,j}$, $u^{i,j,\ell}$ and $t^{i,j,\ell}$ and checks if the total error is non-trivial in which case the simulator aborts the algorithm. From Theorem 9.3 we know that in case there is an error on the $u^{i,j,\ell}$, the algorithm always aborts thus this simulation is perfect.
- For each fault as in group (4). Similar to the previous step, the simulator checks the total error on both $u^{i,j,1}$ and $u^{i,j,\ell}$. Since the simulator is given both $a^{i,1}$ and $b^{j,1}$, it can perfectly simulate whether the algorithm aborts.
- For each fault as in group (5), the simulator does not abort.

From Step 1 of the simulation of the abort signal we see that the simulation fails with a maximal probability of $1/|\mathbb{F}_2|^{d+1}$ for a fault in group (1). Since the simulator aborts in case such a fault occurs, the overall probability for the simulation to fail is $1/|\mathbb{F}_2|^{d+1}$ which can be made negligible in d.

Since $|I| \leq d_1 + k_1 + k_2$ and $|J| \leq d_1 + k_1 + k_2$ and since the abort signal and the probed variables are simulated using only the knowledge of $\{a^{i,1}\}_{i \in I}$ and $\{b^{i,1}\}_{i \in J}$, Algorithm 9.1 is (d, k)-SNINA. □

9.6 The Tile Model

In the previous section, we explored a composable combined security notion called NINA. However, the combined adversary model described in Section 9.4 is quite standard and does not capture many realistic fault attacks. In this section, we explore a method where the implementation is split in multiple tiles. We show that this method allows for an extension of the active adversary to capture more realistic faults attacks. We wrap up by outlining a multiplier which is secure in this tile model.

9.6.1 Definition

The tile probe-and-fault model, as introduced by Reparaz *et al.* [89], captures implementations which are segmented into several areas called tiles. This tile model is linked to Multi-Party Computation (MPC) where each tile would represent a party. A tile is defined as follows.

Definition 9.7 (Tile)**.** A *tile* is a set of hardware resources (wiring, logic, and potential RNGs) dividing the platform where the tiles are interconnected by wires.

The redefinition of the circuit model then captures how to represent algorithms in a tiled construction.

Definition 9.8 (Tiled circuit)**.** A *tiled circuit* is a directed acyclic graph where each node is a resource with a tag indicating which tile it belongs to and which clock cycles activate it.

The tiled methodology also considers a passive adversary as one capable of observing segments of the tiled circuit.

Definition 9.9 (Transient tile probe)**.** A *transient tile probe* observes all resources of a tile which were used for computation during a single clock cycle.

A transient tile fault on the other hand allows the adversary to fault all computation in a tile.

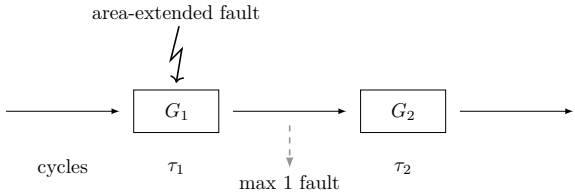


Figure 9.3: The effect of an area-extended fault on two sequential circuits G_1, G_2 where both circuits are 1-SNA considering area-extended faults. The first circuit G_1 runs the first τ_1 cycles followed by τ_2 cycles of circuit G_2. At most 1 output of G_1 can be faulted.

Definition 9.10 (Transient tile fault)**.** A *tile fault* faults (bit-flip or stuck-at) all resources of a tile during a single clock cycle.

Finally, (d, k)-order combined security considers the circuits correctness and privacy when up to d tiles are corrupted, at most k of which being faulted.

Composable security for the tiled methodology follows standard definitions from the field of multi-party computation. More specifically, it is sufficient to consider stand-alone security with abort in the static model. Since sequential operations represent different samples on a power trace, this security model is sufficient. It is formally shown by Canetti [22] that proving security under the stand-alone definitions for secure multi-party computation suffices for obtaining security under sequential general composition.

9.6.2 Area Faults

Typically laser or EM pulses are capable of affecting entire areas in an implementation instead of a single wire. For this fault to be useful for a cryptographic attack, this area is assumed to be limited.

Wire Fault Model To protect against area-wide faults, one considers a topological circuit where each pair of wires has the label of their distance and whether they are coupled or not. Area-extended faults are defined as faults which affect all wires within a certain distance of each other.

Definition 9.11 (Area-extended fault)**.** For any set of adjacent wires $W = (w_1, ..., w_k)$, area-extended faults can be modelled with *c-extended wire faults* so that faulting one wire w_i allows the adversary to fault c wires adjacent to w_i.

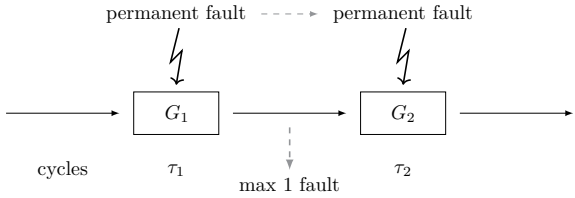


Figure 9.4: The effect of a permanent wire fault on two sequential circuits G_1, G_2 where both circuits are 1-SNA considering permanent wire faults. The first circuit G_1 runs the first τ_1 cycles followed by τ_2 cycles of circuit G_2. In case G_1 and G_2 share the same faulted resource, it is possible for G_2 to receive a faulted input while also being affected by the permanent fault.

The non-accumulation definitions can be adapted to consider area-extended faults as opposed to normal wire faults. Using this extension, noting that area faults affect only parallel running circuits, k-SNA security is sufficient for composable security against k area-extended faults. A visual representation is given in Figure 9.3.

Tile Fault Model Considering the tile model, by requiring that all resources belonging to the same tile are bundled together and assuming that an area-extended fault only affects one tile (which was also assumed in the work by Reparaz *et al.* [89]), area-extended faults are modelled by tile faults. This reduces the area fault security for a complex circuit to the assumption that the fault targets one tile. As a result, the stand-alone model with combined corruptions is sufficient for composable security considering area-extended faults.

9.6.3 Permanent Faults

Consider fault injections which target specific resources in the platform and fault them throughout the entire operation (think of faulting the output of an RNG to zero due to a dedicated laser).

Wire Fault Model Permanent wire faults are defined as those which target a certain hardware resource faulting it for the entire operation. Circuits are now viewed with cycles to model that resources are used more than once in the computation.

Definition 9.12 (Permanent wire fault)**.** For any wire w, faults on all values passing through w can be modelled with *r-extended faults* so that faulting one wire allows the adversary to fault all r values passing through that wire.

Consider the extension of the non-accumulation model considering permanent wire faults. In this case, however, an overhead on the security is required meaning that that each circuit is required to be $2k$-SNA to secure against k permanent wire faults. Requiring that each circuit is $2k$-SNA is sufficient since each circuit gives back at most k faulty shares and each circuit is faulted at most k times. A visual representation is given in Figure 9.4.

Tile Fault Model To model adversaries controlling resources for a longer period in a tiled structure, tile faults are considered as those which affect a tile during the entire operation. Thus, analogous to the definition of a permanent tile probe, we define a permanent tile fault.

Definition 9.13 (Permanent tile fault)**.** A *permanent tile fault* faults all resources of a tile during the entire run of the protocol.

It is clear that a permanent tile fault is stronger than a permanent wire fault as the tile fault affects more than one resource in the circuit. Due to the stand-alone model capturing the security in MPC protocols where adversaries corrupt a party for the entire duration of the computation, this model is sufficient for composable security considering the permanent faults.

9.6.4 A Combined Secure Tiled Multiplier

We wrap up by describing a multiplier which is combined secure in the tile model. For ease, we only describe a first-order version of the multiplier, meaning $d = k = 1$. The reader is referred to the full paper for the general algorithm and security proof [49].

Security with abort is considered where each tile keeps a copy of the abort flag. The used sharing is a duplicated Boolean sharing where each value is shared in two masks and each mask is duplicated two times. This countermeasure protects against a combined adversary which either observes or faults one tile. There are a total of eight tiles where four are main tiles $T^{i,t}$ and four are auxiliary tiles $T^{i,j,t}$. The tiles $T^{1,1}, T^{2,1}$ have random bit generators, and $T^{i,j,t}$ hold abort states such that when $T^{i,j,t}$ aborts the other tiles halt as well.

A depiction of the multiplier is given in Figure 9.5. The multiplication operates by sending all each duplicate share from the main tiles $T^{i,t}$ to the auxiliary

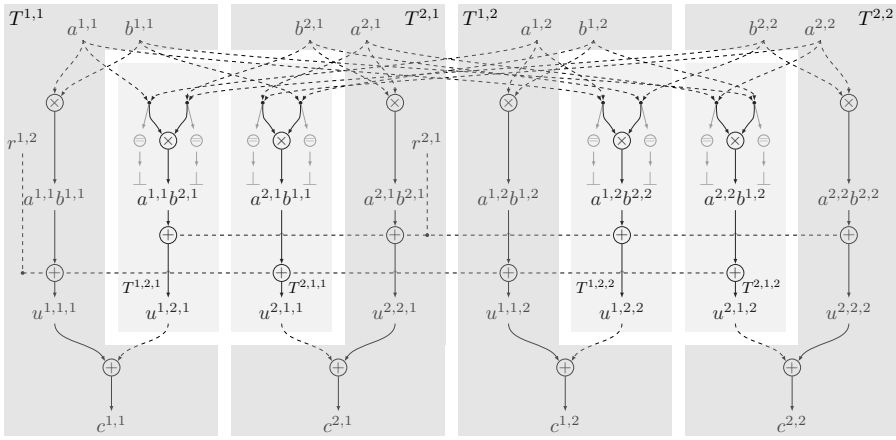


Figure 9.5: The combined secure tiled multiplier over $\mathbb{F}_{2^m}$ with $d = 1$ and $k = 1$, where dashed lines denote values taken from registers. The auxiliary tiles $T^{i,j,t}$ receive all duplicate shares $a^{i,\ell}$ and $b^{j,\ell}$ for $\ell \in \{1, 2\}$ and check their integrity.

tiles $T^{i,j,t}$ and $T^{j,i,t}$. These auxiliary tiles then perform an error check on the received duplicates. Randomness is then sampled in the main tiles and sent to the auxiliary tiles to refresh the cross products. In case no errors are found, the auxiliary tiles $T^{i,j,t}$ send their cross product back to the main tiles. Last, the main tiles add the received cross products to obtain a share of the multiplication.

9.7 Conclusion

In this chapter, we introduced standard adversary models for fault attacks and combined attacks. We then proposed composable security models, non-accumulation and NINA, which guarantee security against these fault and combined adversaries. In the previous chapters, we have seen that non-interference models typically incur a high randomness cost. The non-accumulation and NINA models also demand the same cost. Moreover, orthogonal to randomness costs for masking methods, NINA-secured encodings will typically require several error checks on a gate level. These checks require a non-negligible combinatorial cost and the ability to abort the computation which is not trivial to ensure. As such, we believe there is merit in further improving the efficiency of NINA secured circuits or the security model itself.

We also introduced the tile model in which implementations are split into

multiple parts. The adversary model linked to the tile model is particularly strong as the adversary can observe or faults entire segments of the computation with full control. We proposed extensions to the standard fault adversary in two directions. The first is to allow for a fault to affect more than one resource in order to model area faults. The second is to allow for permanent faults where the adversary has the capability to fault the same resource multiple times in order to better model laser fault injections. We then discussed the extension's effect in the NINA security model and we argued that the tile model captures the extensions.

For both the NINA model and the tile model, we proposed a multiplier secure in the respective security models. These multipliers can be used in a methodology to secure any computation.

Chapter 10

Conclusions & Open Problems

In this chapter, we provide a summary of the thesis and look back at the research directions posed in Section 1.2. We also look forward and pose some open problems for future research.

10.1 Summary

In this thesis, we tackled two main research directions. The randomness usage of masking schemes and security models capturing active and combined adversaries.

Higher-Order Threshold Implementations We first analysed the randomness usage of maskings. We found that many research articles did not report on the cost or method of the generation of randomness needed to secure the masking. Conversely, there were several articles reporting using non-cryptographic methods to generate this randomness. We gave arguments to support that these non-cryptographic methods might lead to significant reductions in the masking's security. At the very least, it shows a security analysis for the random number generator should be made in complement to the argument for the masking. We also showed that the cryptographic secure generation of randomness can come at a high cost. The research question becomes how we can make maskings which are secure and yet do not rely on a heavy random number generator.

We investigate a solution proposed in 2006, called *threshold implementations*. The solution has been very popular because of its use in hardware maskings. However, we find that the property of *uniformity* aids the research question posed above. We show that threshold implementations include maskings which require no fresh randomness as opposed to the sharings secure in the non-interference model. To show the utility of threshold implementations, we transform a popular efficient first-order sharing of the AES into one that requires no fresh randomness at almost no additional cost. Threshold implementations thus solve the issue of randomness for first-order security. However, these maskings do not offer security against higher-order attacks.

We propose a new security model called the *bounded-query probing model* to release the requirement of perfect security. We then provide an analysis technique based on linear cryptanalysis to show whether maskings are secure in this new security model. This solves the long-standing open problem on how to analyse the security of higher-order threshold implementations. These higher-order maskings have the same advantages as the original threshold implementations. We show that we can again create maskings which require no fresh randomness and are higher-order secure. Nevertheless, the maskings still face several engineering challenges. The theory can provide upper bounds on the advantage of the probing adversary, but these bounds can in some cases be too high to conclude practical security. These higher-order threshold implementations can also be quite inefficient as they require many shares or the symmetric primitive does not offer usable cryptanalytic properties. To this end, we have faced these engineering challenges by proposing an extension to the probing model called *the noisy probing model* and providing new masking techniques which allow for the reduction of the number of shares and register stages. We also proposed a low-randomness second-order masked AES. This has resulted in an improvement in the glitch model and improved diffusion properties from the changing of the guards method. Finally, we used the cryptanalytic security analysis to argue whether maskings are secure against advanced leakage effects such as transition leakage or coupling leakage. We found that protecting maskings against these effects can often be done without significant overhead.

The end result of this line of research is that we can analyse the security of masking schemes including the generation of randomness and use this analysis to significantly reduce the needed randomness. This creates a significant improvement in the efficiency of maskings, the trade-off being the reduction from perfect security to bounded-query security. This also improves the trust in maskings as a reduced randomness cost means a reduced reliance on a secure component. The academic novelty lies in the relation between side-channel analysis and linear cryptanalysis and the laying of the foundation of higher-order threshold implementations.

Active and Combined Security Models In the second line of research we created adversary and security models for fault and combined attacks. We proposed the active variant of the probing adversary which can control a threshold number of resources in a rolled-out circuit. The combined adversary being the combination of the probing adversary with this fault adversary. In order to create circuits which are secure against these new adversaries, we expanded the non-interference framework to its active counterpart, called the *non-accumulation* model. This model is then combined with the non-interference framework to create the notion of NINA security. We provided a multiplication operation which is secure in this new model to show it provides the same functionality of the non-interference framework. The secure multiplication operation together with an addition operation creates a methodology to secure arbitrary computation.

We then continued this research by considering more advanced adversaries which better relate to reality. However, as these adversaries have an increased impact on circuits, better security models were needed to secure against them. We provided a connection to a multiparty computation-like adversary which captures all the fault extensions. As a result, we were capable of creating secure operations against the improved fault attacks with the only overhead being in the fresh randomness of the computation. The academic contribution is the deepening of the link between multiparty computation and physical security.

10.2 Directions for Future Work

In our research, we encountered several interesting open problems. Here we summarise the main ones.

1: Number of Traces In the current lab practice, maskings are verified against 100 million traces. This number has organically grown over the years and has manifested from the limits of data processing from ports in FPGAs or oscilloscopes and other factors such as memory limitations or the clock frequency of the application. While performing lab practices using 100 million traces is not an issue, researchers and companies have started to confuse the limitations of a lab with the limitations of an adversary. It is not evident to assume an adversary can not gather more than 100 million traces. It certainly is not evident to only research countermeasures which provide security for that level of data.

In symmetric cryptography the data limitation is based on computational limitations. Typical security levels are 80, 128, or 256 bits, this should be

compared with the 27 bit data level in side-channel analysis. In order to relate the security level with practical security, the National Institute of Standards and Technology (NIST) provides recommendations every couple of years. The security level in symmetric cryptography accounts for all possible attack vectors, meaning an adversary having access to multiple (billions) devices or the influence of precomputation or parallel computation. Even quantum computation is taken into account in some cases. Compare this to side-channel analysis where the data complexity has not been researched. We have no justification for the bottle neck on the number of traces an adversary can take. That means that it is possible that via a different attack method or via different equipment in the lab, it becomes possible to take more traces. Meaning that large increases in the data complexity can not be ruled out. Such an improvement would be nothing short from devastating to the field as suddenly the need for extreme high orders of protection would be required from the masking. Such high orders might be too inefficient a solution in practice.

In order to spark the creativity of the reader, such a significant improvement in the data complexity could arise from a software-bug/API-feature in an application on a phone where an app could ask for its, or other application's, specific power consumption. Given that the battery level of a phone can be retrieved quite accurately, it seems possible to accurately portrait this power consumption. Using such a "feature", attackers would easily recover a high number of traces over many phones.

We thus believe that masking methods should be made with the idea that attackers could do better than the current lab practices. Perhaps by combining masking with other countermeasures, such as limiting the data available to the attacker, to achieve high levels of protection without increasing the overhead too much.

2: A More Realistic Adversary Model While the probing model was invaluable to the creation of secure maskings as we know them, the model has its obvious limitations. In some aspects, the probing model is too weak. For example, think of glitch or transition effects described in Section 2.5.3 and Section 2.5.4. In other cases, the probing model is too strong. For example, the adversary never observes the exact value on a wire.

Research should be invested to bring the probing model closer to reality. For example, with the noisy probing model from Chapter 6 the adversary receives a noisy value instead of the exact value on a wire. But this model could be extended further. The adversary should not be limited in the number of probes as additional probes might not improve the adversary's advantage due to the noise amplification. The model should also be expanded to include the effects

of glitches, transition leakage, and parallel computation and make the noise dependent on the operation which is performed (e.g. a heavy S-box versus a single XOR). Afterwards, a good analysis method should be made to easily verify the security of maskings. Meaning that the new security model should not loose the current advantages of the probing model.

Such a realistic model is key to progressing research as it allows for cryptographic bounds of maskings. Meaning that we should be able to predict the security bound of a masking given the noise of the platform and measuring equipment. The current lab analysis only shows whether a masking is secure under 100 million traces. But it does not predict whether it will be secure under 200 million traces. It does not quantify security and this is a key aspect in the field of cryptography.

3: Creating a Random Number Generator for Maskings In this thesis, we explored how to reduce the randomness requirements of maskings. However, the strong non-interference framework has an advantage over the security analysis in the bounded-query model. The solutions found the non-interference framework allow for arbitrary composition. Apart from securing symmetric primitives, secure solutions typically need to mask various other operations outside the primitive such as data processing. Performing an intricate analysis for each operation is cumbersome. Instead, the costs involving randomness generation could be reduced by proposing a lightweight random number generator specifically tailored to the needs of masking. This means proposing a stream cipher secure against a very limited cryptanalytic adversary. From what we currently know now, the stream cipher's state is typically refreshed with true randomness every masked encryption call. In that masked encryption call, typically some thousand to ten thousand random bits are required. That means we can consider an adversary with at most 80-bit computational complexity and 2^{14} calls to the stream cipher. Further study in the field of masking could further limit the adversary. Given that current lightweight stream ciphers often consider an adversary which can interact up to 2^{80} times with the primitive, this small adaptation could already lead to a good improvement in the cost of randomness generation.

4: Masking-Friendly Primitives The research in this thesis reveals extra properties maskings can have to improve their security. We have seen that the diffusion of the primitive and the confusion of the masked S-box play important roles to protect against higher-order attacks or attacks using more advanced leakage functions like Hamming distance leakage. An open problem in this research is the culmination of the findings into a masking-friendly symmetric

primitive. While this work has put several steps into this direction, there are still many unsolved questions. We have studied the nonlinearity of a masked S-box, but have given no link to the nonlinearity of the S-box proper. Perhaps research can uncover bounds linking one to the other, or specific constructions which maximise the property. We have also observed masked S-boxes can have diffusion properties, for example masked S-boxes exhibiting nontrivial branch numbers over their masked input. It would be interesting to see whether efficient masking of known S-boxes exist which have a nontrivial branch number. Finally, side-channel adversaries can only accurately view small parts of the computation at a time due to the noise. As a result, slow diffusion in the sense of long zero-correlation approximations through the symmetric primitive becomes important. A heuristic in this sense is how many S-boxes are activated in the shortest activity pattern which connects one S-box to another. Some numbers are given in Table 10.1. In this table we can see some diffusion layers can perform better than others. We believe these insight can help to create a masking-friendly primitive which can outperform the current standards.

Table 10.1: Minimal activity patterns starting and ending in a single S-box using generic S-boxes.

Primitive	State Size	S-Box Size	# Active S-Boxes
PRESENT	64 bits	4 bits	2
PRINCE	64 bits	4 bits	6
SHARK [91]	64 bits	8 bits	10
AES	128 bits	8 bits	26
ASCON	320 bits	5 bits	61
Keccak-f[400]	400 bits	5 bits	76

5: Active Security with Minimal Aborts The main goal of the research was to minimise the randomness cost of makings and to investigate active security. The two research goals could be combined. The masked multipliers given in Chapter 9 perform error checks in order to protect against injected faults. This means that over an encoded cipher, several tens-of-thousand error checks are performed. Moreover, there is a need for an abort system to immediately halt the computation when a fault is detected. This mechanism should in itself also protect against fault attacks making it very expensive to implement.

The work by Daemen *et al.* [36] explains a method to protect encodings against fault attacks, and statistical ineffective fault attacks, and only error check the output of the encoded primitive. This method thus not only saves in the number of error checks, it does not require an expensive abort mechanism. However, the work only protects against one fault. This is reminiscent of the limitation

of threshold implementations. Thus, we pose a future line of research to extend the work by Daemen *et al.* by using cryptanalytic methods to protect against multiple faults.

Bibliography

[1] Ågren, M., Hell, M., Johansson, T., and Meier, W. Grain-128a: a new version of grain-128 with optional authentication. *Int. J. Wirel. Mob. Comput. 5*, 1 (2011), 48–59.

[2] Balasch, J., Faust, S., Gierlichs, B., Paglialonga, C., and Standaert, F.-X. Consolidating inner product masking. In *Advances in Cryptology – ASIACRYPT 2017, Part I* (Hong Kong, China, Dec. 3–7, 2017), T. Takagi and T. Peyrin, Eds., vol. 10624 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 724–754.

[3] Banik, S., Pandey, S. K., Peyrin, T., Sasaki, Y., Sim, S. M., and Todo, Y. GIFT: A small present - towards reaching the limit of lightweight encryption. In *Cryptographic Hardware and Embedded Systems – CHES 2017* (Taipei, Taiwan, Sept. 25–28, 2017), W. Fischer and N. Homma, Eds., vol. 10529 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 321–345.

[4] Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.-A., Grégoire, B., Strub, P.-Y., and Zucchini, R. Strong non-interference and type-directed higher-order masking. In *ACM CCS 2016: 23rd Conference on Computer and Communications Security* (Vienna, Austria, Oct. 24–28, 2016), E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds., ACM Press, pp. 116–129.

[5] Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F.-X., and Strub, P.-Y. Parallel implementations of masking schemes and the bounded moment leakage model. In *Advances in Cryptology – EUROCRYPT 2017, Part I* (Paris, France, Apr. 30 – May 4, 2017), J. Coron and J. B. Nielsen, Eds., vol. 10210 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 535–566.

[6] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., and Sim, S. M. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology – CRYPTO 2016, Part II* (Santa Barbara, CA, USA, Aug. 14–18, 2016), M. Robshaw and J. Katz, Eds., vol. 9815 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 123–153.

[7] Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., and Vergnaud, D. Randomness complexity of private circuits for multiplication. In *Advances in Cryptology – EUROCRYPT 2016, Part II* (Vienna, Austria, May 8–12, 2016), M. Fischlin and J.-S. Coron, Eds., vol. 9666 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 616–648.

[8] Beyne, T. Block cipher invariants as eigenvectors of correlation matrices. In *Advances in Cryptology – ASIACRYPT 2018, Part I* (Brisbane, Queensland, Australia, Dec. 2–6, 2018), T. Peyrin and S. Galbraith, Eds., vol. 11272 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 3–31.

[9] Beyne, T. Linear cryptanalysis in the weak key model. Master Thesis, KU Leuven, 2019.

[10] Beyne, T., Dhooghe, S., Moradi, A., and Shahmirzadi, A. R. Cryptanalysis of efficient masked ciphers: Applications to low latency. Cryptology ePrint Archive, Report 2021/1378, 2021. `https://ia.cr/2021/1378`.

[11] Beyne, T., Dhooghe, S., Ranea, A., and Sijacic, D. A low-randomness second-order masked AES. *IACR Cryptol. ePrint Arch.* (2021), 1111.

[12] Beyne, T., Dhooghe, S., and Zhang, Z. Cryptanalysis of masked ciphers: A not so random idea. In *Advances in Cryptology – ASIACRYPT 2020, Part I* (Dec. 2020), Lecture Notes in Computer Science, Springer, Heidelberg, Germany, pp. 817–850.

[13] Biham, E., and Shamir, A. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology – CRYPTO'97* (Santa Barbara, CA, USA, Aug. 17–21, 1997), B. S. Kaliski Jr., Ed., vol. 1294 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 513–525.

[14] Bilgin, B. *Threshold implementations : as countermeasure against higher-order differential power analysis*. PhD thesis, University of Twente, Enschede, Netherlands, 2015.

[15] Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., and Rijmen, V. Higher-order threshold implementations. In *Advances in Cryptology – ASIACRYPT 2014, Part II* (Kaoshiung, Taiwan, R.O.C., Dec. 7–11, 2014), P. Sarkar and T. Iwata, Eds., vol. 8874 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 326–343.

[16] Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., and Rijmen, V. Trade-offs for threshold implementations illustrated on AES. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 34*, 7 (2015), 1188–1200.

[17] Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., and Stütz, G. Threshold implementations of all 3×3 and 4×4 S-boxes. In *Cryptographic Hardware and Embedded Systems – CHES 2012* (Leuven, Belgium, Sept. 9–12, 2012), E. Prouff and P. Schaumont, Eds., vol. 7428 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 76–91.

[18] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J. B., Seurin, Y., and Vikkelsoe, C. PRESENT: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems – CHES 2007* (Vienna, Austria, Sept. 10–13, 2007), P. Paillier and I. Verbauwhede, Eds., vol. 4727 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 450–466.

[19] Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E. B., Knežević, M., Knudsen, L. R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S. S., and Yalçın, T. PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In *Advances in Cryptology – ASIACRYPT 2012* (Beijing, China, Dec. 2–6, 2012), X. Wang and K. Sako, Eds., vol. 7658 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 208–225.

[20] Bozilov, D. On optimality of d + 1 TI shared functions of 8 bits or less. *IACR Cryptol. ePrint Arch. 2020* (2020), 570.

[21] Braeken, A., Nikov, V., Nikova, S., and Preneel, B. On boolean functions with generalized cryptographic properties. In *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings* (2004), A. Canteaut and K. Viswanathan, Eds., vol. 3348 of *Lecture Notes in Computer Science*, Springer, pp. 120–135.

[22] Canetti, R. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology 13*, 1 (Jan. 2000), 143–202.

[23] Cannière, C. D. Trivium: A stream cipher construction inspired by block cipher design principles. In *ISC* (2006), vol. 4176 of *Lecture Notes in Computer Science*, Springer, pp. 171–186.

[24] Canright, D. A very compact S-box for AES. In *Cryptographic Hardware and Embedded Systems – CHES 2005* (Edinburgh, UK, Aug. 29 – Sept. 1, 2005), J. R. Rao and B. Sunar, Eds., vol. 3659 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 441–455.

[25] Carlet, C., Crama, Y., and Hammer, P. L. Boolean functions for cryptography and error correcting codes. *Boolean models and methods in mathematics, computer science, and engineering 2* (2010), 257–397.

[26] Cassiers, G., Grégoire, B., Levi, I., and Standaert, F.-X. Hardware private circuits: From trivial composition to full verification. Cryptology ePrint Archive, Report 2020/185, 2020. `https://eprint.iacr.org/2020/185`.

[27] Cassiers, G., and Standaert, F.-X. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Trans. Inf. Forensics Secur. 15* (2020), 2542–2555.

[28] Cassiers, G., and Standaert, F.-X. Provably secure hardware masking in the transition- and glitch-robust probing model: Better safe than sorry. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2021*, 2 (2021), 136–158. `https://tches.iacr.org/index.php/TCHES/article/view/8790`.

[29] Castagnos, G., Renner, S., and Zémor, G. High-order masking by using coding theory and its application to AES. In *14th IMA International Conference on Cryptography and Coding* (Oxford, UK, Dec. 17–19, 2013), M. Stam, Ed., vol. 8308 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 193–212.

[30] Chari, S., Jutla, C. S., Rao, J. R., and Rohatgi, P. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology – CRYPTO'99* (Santa Barbara, CA, USA, Aug. 15–19, 1999), M. J. Wiener, Ed., vol. 1666 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 398–412.

[31] Clavier, C. Secret external encodings do not prevent transient fault analysis. In *Cryptographic Hardware and Embedded Systems – CHES 2007* (Vienna, Austria, Sept. 10–13, 2007), P. Paillier and I. Verbauwhede, Eds., vol. 4727 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 181–194.

[32] Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., and Verneuil, V. Horizontal correlation analysis on exponentiation. In *ICICS 10: 12th International Conference on Information and Communication Security* (Barcelona, Spain, Dec. 15–17, 2010), M. Soriano, S. Qing, and J. López, Eds., vol. 6476 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 46–61.

[33] Coron, J.-S., Prouff, E., Rivain, M., and Roche, T. Higher-order side channel security and mask refreshing. In *Fast Software Encryption – FSE 2013* (Singapore, Mar. 11–13, 2014), S. Moriai, Ed., vol. 8424 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 410–424.

[34] Daemen, J. Spectral characterization of iterating lossy mappings. Cryptology ePrint Archive, Report 2016/090, 2016. `https://eprint.iacr.org/2016/090`.

[35] Daemen, J. Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing. In *Cryptographic Hardware and Embedded Systems – CHES 2017* (Taipei, Taiwan, Sept. 25–28, 2017), W. Fischer and N. Homma, Eds., vol. 10529 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 137–153.

[36] Daemen, J., Dobraunig, C., Eichlseder, M., Gross, H., Mendel, F., and Primas, R. Protecting against statistical ineffective fault attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2020*, 3 (2020), 508–543. `https://tches.iacr.org/index.php/TCHES/article/view/8599`.

[37] Daemen, J., Govaerts, R., and Vandewalle, J. Correlation matrices. In *Fast Software Encryption – FSE'94* (Leuven, Belgium, Dec. 14–16, 1995), B. Preneel, Ed., vol. 1008 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 275–285.

[38] Daemen, J., and Rijmen, V. The block cipher Rijndael. In *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings* (1998), J. Quisquater and B. Schneier, Eds., vol. 1820 of *Lecture Notes in Computer Science*, Springer, pp. 277–284.

[39] Daemen, J., and Rijmen, V. The wide trail design strategy. In *8th IMA International Conference on Cryptography and Coding* (Cirencester, UK, Dec. 17–19, 2001), B. Honary, Ed., vol. 2260 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 222–238.

[40] Daemen, J., and Rijmen, V. *The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition.* Information Security and Cryptography. Springer, 2020.

[41] De Cnudde, T., Bilgin, B., Gierlichs, B., Nikov, V., Nikova, S., and Rijmen, V. Does coupling affect the security of masked implementations? In *COSADE 2017: 8th International Workshop on Constructive Side-Channel Analysis and Secure Design* (Paris, France, Apr. 13–14, 2017), S. Guilley, Ed., vol. 10348 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 1–18.

[42] De Cnudde, T., Ender, M., and Moradi, A. Hardware masking, revisited. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2018*, 2 (2018), 123–148. https://tches.iacr.org/index.php/TCHES/article/view/877.

[43] De Cnudde, T., Reparaz, O., Bilgin, B., Nikova, S., Nikov, V., and Rijmen, V. Masking AES with d+1 shares in hardware. In *Cryptographic Hardware and Embedded Systems – CHES 2016* (Santa Barbara, CA, USA, Aug. 17–19, 2016), B. Gierlichs and A. Y. Poschmann, Eds., vol. 9813 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 194–212.

[44] De Meyer, L. Recovering the CTR_DRBG state in 256 traces. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2020*, 1 (2019), 37–65. https://tches.iacr.org/index.php/TCHES/article/view/8392.

[45] De Meyer, L., Arribas, V., Nikova, S., Nikov, V., and Rijmen, V. M&M: Masks and macs against physical attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2019*, 1 (2018), 25–50. https://tches.iacr.org/index.php/TCHES/article/view/7333.

[46] De Meyer, L., Bilgin, B., and Reparaz, O. Consolidating security notions in hardware masking. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2019*, 3 (2019), 119–147. https://tches.iacr.org/index.php/TCHES/article/view/8291.

[47] De Meyer, L., Reparaz, O., and Bilgin, B. Multiplicative masking for AES in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2018*, 3 (2018), 431–468. https://tches.iacr.org/index.php/TCHES/article/view/7282.

[48] Dhooghe, S. Analyzing masked ciphers against transition and coupling effects. *IACR Cryptol. ePrint Arch.* (2021), 1095.

[49] Dhooghe, S., and Nikova, S. Let's tessellate: Tiling for security against advanced probe and fault adversaries. In *CARDIS* (2020), vol. 12609 of *Lecture Notes in Computer Science*, Springer, pp. 181–195.

[50] Dhooghe, S., and Nikova, S. My gadget just cares for me - how NINA can prove security against combined attacks. In *Topics in Cryptology – CT-RSA 2020* (San Francisco, CA, USA, Feb. 24–28, 2020), S. Jarecki, Ed., vol. 12006 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 35–55.

[51] Dhooghe, S., and Nikova, S. Resilient uniformity: Applying resiliency in masking. *IACR Cryptol. ePrint Arch.* (2021), 1094.

[52] Dhooghe, S., Nikova, S., and Rijmen, V. Threshold implementations in the robust probing model. In *TIS@CCS* (2019), ACM, pp. 30–37.

[53] Dhooghe, S., Nikova, S., and Rijmen, V. First-order hardware sharings of the AES. *IACR Cryptol. ePrint Arch. 2021* (2021), 734.

[54] Diaconis, P. *Group representations in probability and statistics*, vol. 11 of *Lecture Notes–Monograph Series*. Institute of Mathematical Statistics, Hayward, CA, 1988.

[55] Dobraunig, C., Eichlseder, M., Korak, T., Mangard, S., Mendel, F., and Primas, R. SIFA: Exploiting ineffective fault inductions on symmetric cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2018*, 3 (2018), 547–572. `https://tches.iacr.org/index.php/TCHES/article/view/7286`.

[56] Dobraunig, C., Eichlseder, M., Mendel, F., and Schläffer, M. Ascon v1.2. https://ascon.iaik.tugraz.at/files/asconv12-nist.pdf.

[57] Dziembowski, S., Faust, S., Herold, G., Journault, A., Masny, D., and Standaert, F.-X. Towards sound fresh re-keying with hard (physical) learning problems. In *Advances in Cryptology – CRYPTO 2016, Part II* (Santa Barbara, CA, USA, Aug. 14–18, 2016), M. Robshaw and J. Katz, Eds., vol. 9815 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 272–301.

[58] Dziembowski, S., and Pietrzak, K. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science* (Philadelphia, PA, USA, Oct. 25–28, 2008), IEEE Computer Society Press, pp. 293–302.

[59] Faust, S., Grosso, V., Pozo, S. M. D., Paglialonga, C., and Standaert, F.-X. Composable masking schemes in the presence of

physical defaults & the robust probing model. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2018*, 3 (2018), 89–120. https://tches.iacr.org/index.php/TCHES/article/view/7270.

[60] Goubin, L., and Martinelli, A. Protecting AES with Shamir's secret sharing scheme. In *Cryptographic Hardware and Embedded Systems – CHES 2011* (Nara, Japan, Sept. 28 – Oct. 1, 2011), B. Preneel and T. Takagi, Eds., vol. 6917 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 79–94.

[61] Goubin, L., and Patarin, J. DES and differential power analysis (the "duplication" method). In *Cryptographic Hardware and Embedded Systems – CHES'99* (Worcester, Massachusetts, USA, Aug. 12–13, 1999), Çetin Kaya. Koç and C. Paar, Eds., vol. 1717 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 158–172.

[62] Gross, H., Mangard, S., and Korak, T. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016* (2016), B. Bilgin, S. Nikova, and V. Rijmen, Eds., ACM, p. 3.

[63] Gross, H., Mangard, S., and Korak, T. An efficient side-channel protected AES implementation with arbitrary protection order. In *Topics in Cryptology – CT-RSA 2017* (San Francisco, CA, USA, Feb. 14–17, 2017), H. Handschuh, Ed., vol. 10159 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 95–112.

[64] Guo, J., Peyrin, T., and Poschmann, A. The PHOTON family of lightweight hash functions. In *Advances in Cryptology – CRYPTO 2011* (Santa Barbara, CA, USA, Aug. 14–18, 2011), P. Rogaway, Ed., vol. 6841 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 222–239.

[65] Guo, J., Peyrin, T., Poschmann, A., and Robshaw, M. J. B. The LED block cipher. In *Cryptographic Hardware and Embedded Systems – CHES 2011* (Nara, Japan, Sept. 28 – Oct. 1, 2011), B. Preneel and T. Takagi, Eds., vol. 6917 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 326–341.

[66] Hirt, M., and Maurer, U. M. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *16th ACM Symposium Annual on Principles of Distributed Computing* (Santa Barbara, CA, USA, Aug. 21–24, 1997), J. E. Burns and H. Attiya, Eds., Association for Computing Machinery, pp. 25–34.

[67] Ishai, Y., Kushilevitz, E., Li, X., Ostrovsky, R., Prabhakaran, M., Sahai, A., and Zuckerman, D. Robust pseudorandom generators. In *ICALP 2013: 40th International Colloquium on Automata, Languages and Programming, Part I* (Riga, Latvia, July 8–12, 2013), F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, Eds., vol. 7965 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 576–588.

[68] Ishai, Y., Prabhakaran, M., Sahai, A., and Wagner, D. Private circuits II: Keeping secrets in tamperable circuits. In *Advances in Cryptology – EUROCRYPT 2006* (St. Petersburg, Russia, May 28 – June 1, 2006), S. Vaudenay, Ed., vol. 4004 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 308–327.

[69] Ishai, Y., Sahai, A., and Wagner, D. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology – CRYPTO 2003* (Santa Barbara, CA, USA, Aug. 17–21, 2003), D. Boneh, Ed., vol. 2729 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 463–481.

[70] Kalai, Y. T., and Reyzin, L. A survey of leakage-resilient cryptography. Cryptology ePrint Archive, Report 2019/302, 2019. `https://eprint.iacr.org/2019/302`.

[71] Knichel, D., Sasdrich, P., and Moradi, A. SILVER - statistical independence and leakage verification. In *Advances in Cryptology – ASIACRYPT 2020, Part I* (Dec. 2020), Lecture Notes in Computer Science, Springer, Heidelberg, Germany, pp. 787–816.

[72] Knichel, D., Sasdrich, P., and Moradi, A. Generic hardware private circuits - towards automated generation of composable secure gadgets. *IACR Cryptol. ePrint Arch. 2021* (2021), 247.

[73] Kocher, P. C., Jaffe, J., and Jun, B. Differential power analysis. In *Advances in Cryptology – CRYPTO'99* (Santa Barbara, CA, USA, Aug. 15–19, 1999), M. J. Wiener, Ed., vol. 1666 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 388–397.

[74] Kutzner, S., Nguyen, P. H., Poschmann, A., and Wang, H. On 3-share threshold implementations for 4-bit S-boxes. In *COSADE 2013: 4th International Workshop on Constructive Side-Channel Analysis and Secure Design* (Paris, France, Mar. 6–8, 2013), E. Prouff, Ed., vol. 7864 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 99–113.

[75] Mangard, S., Pramstaller, N., and Oswald, E. Successfully attacking masked AES hardware implementations. In *Cryptographic*

Hardware and Embedded Systems – CHES 2005 (Edinburgh, UK, Aug. 29 – Sept. 1, 2005), J. R. Rao and B. Sunar, Eds., vol. 3659 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 157–171.

[76] Matsui, M. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology – EUROCRYPT'93* (Lofthus, Norway, May 23–27, 1994), T. Helleseth, Ed., vol. 765 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 386–397.

[77] Moos, T., Moradi, A., Schneider, T., and Standaert, F. Glitch-resistant masking revisited or why proofs in the robust probing model are needed. *IACR Trans. Cryptogr. Hardw. Embed. Syst. 2019*, 2 (2019), 256–292.

[78] Moradi, A., Shalmani, M. T. M., and Salmasizadeh, M. A generalized method of differential fault attack against AES cryptosystem. In *Cryptographic Hardware and Embedded Systems – CHES 2006* (Yokohama, Japan, Oct. 10–13, 2006), L. Goubin and M. Matsui, Eds., vol. 4249 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 91–100.

[79] Moradi, A., Shalmani, M. T. M., and Salmasizadeh, M. Dual-rail transition logic: A logic style for counteracting power analysis attacks. *Comput. Electr. Eng. 35*, 2 (2009), 359–369.

[80] Mouha, N., Wang, Q., Gu, D., and Preneel, B. Differential and linear cryptanalysis using mixed-integer linear programming. In *Inscrypt* (2011), vol. 7537 of *Lecture Notes in Computer Science*, Springer, pp. 57–76.

[81] National Institute of Standards and Technology. Recommendation for Random Bit Generator (RBG) Constructions. National Institute of Standards and Technology (NIST), SP 800-90C, April 2016.

[82] National Institute of Standards and Technology. Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, Nov. 2001.

[83] Niemetz, A., Preiner, M., and Biere, A. Boolector 2.0 system description. *Journal on Satisfiability, Boolean Modeling and Computation 9* (2015), 53–58.

[84] Nikova, S., Rechberger, C., and Rijmen, V. Threshold implementations against side-channel attacks and glitches. In *ICICS 06: 8th International Conference on Information and Communication*

Security (Raleigh, NC, USA, Dec. 4–7, 2006), P. Ning, S. Qing, and N. Li, Eds., vol. 4307 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 529–545.

[85] Petrides, G. On non-completeness in threshold implementations. In *Proceedings of ACM Workshop on Theory of Implementation Security Workshop, TIS@CCS 2019, London, UK, November 11, 2019* (2019), B. Bilgin, S. Petkova-Nikova, and V. Rijmen, Eds., ACM, pp. 24–28.

[86] Prouff, E., and Rivain, M. Masking against side-channel attacks: A formal security proof. In *Advances in Cryptology – EUROCRYPT 2013* (Athens, Greece, May 26–30, 2013), T. Johansson and P. Q. Nguyen, Eds., vol. 7881 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 142–159.

[87] Reparaz, O. A note on the security of higher-order threshold implementations. Cryptology ePrint Archive, Report 2015/001, 2015. `https://eprint.iacr.org/2015/001`.

[88] Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., and Verbauwhede, I. Consolidating masking schemes. In *Advances in Cryptology – CRYPTO 2015, Part I* (Santa Barbara, CA, USA, Aug. 16–20, 2015), R. Gennaro and M. J. B. Robshaw, Eds., vol. 9215 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 764–783.

[89] Reparaz, O., De Meyer, L., Bilgin, B., Arribas, V., Nikova, S., Nikov, V., and Smart, N. P. CAPA: The spirit of beaver against physical attacks. In *Advances in Cryptology – CRYPTO 2018, Part I* (Santa Barbara, CA, USA, Aug. 19–23, 2018), H. Shacham and A. Boldyreva, Eds., vol. 10991 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 121–151.

[90] Richter-Brockmann, J., Sasdrich, P., and Güneysu, T. Revisiting fault adversary models - hardware faults in theory and practice. *IACR Cryptol. ePrint Arch. 2021* (2021), 296.

[91] Rijmen, V., Daemen, J., Preneel, B., Bossalaers, A., and De Win, E. The cipher SHARK. In *Fast Software Encryption – FSE'96* (Cambridge, UK, Feb. 21–23, 1996), D. Gollmann, Ed., vol. 1039 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 99–111.

[92] Serre, J.-P. *Linear representations of finite groups*, vol. 42. Springer Science & Business Media, 2012.

[93] Shahmirzadi, A. R., and Moradi, A. Re-consolidating first-order masking schemes. *IACR Transactions on Cryptographic Hardware and*

Embedded Systems 2021, 1 (2021), 305–342. https://tches.iacr.org/index.php/TCHES/article/view/8736.

[94] Shahmirzadi, A. R., and Moradi, A. Second-order SCA security with almost no fresh randomness. *IACR Cryptol. ePrint Arch. 2021* (2021), 461.

[95] Shamir, A. How to share a secret. *Communications of the Association for Computing Machinery 22*, 11 (Nov. 1979), 612–613.

[96] Siegenthaler, T. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Trans. Inf. Theory 30*, 5 (1984), 776–780.

[97] Sugawara, T. 3-share threshold implementation of AES s-box without fresh randomness. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2019*, 1 (2018), 123–145. https://tches.iacr.org/index.php/TCHES/article/view/7336.

[98] Tardy-Corfdir, A., and Gilbert, H. A known plaintext attack of FEAL-4 and FEAL-6. In *Advances in Cryptology – CRYPTO'91* (Santa Barbara, CA, USA, Aug. 11–15, 1992), J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 172–181.

[99] Veyrat-Charvillon, N., Medwed, M., Kerckhof, S., and Standaert, F.-X. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In *Advances in Cryptology – ASIACRYPT 2012* (Beijing, China, Dec. 2–6, 2012), X. Wang and K. Sako, Eds., vol. 7658 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 740–757.

[100] Wegener, F., De Meyer, L., and Moradi, A. Spin me right round rotational symmetry for FPGA-specific AES: Extended version. *Journal of Cryptology 33*, 3 (July 2020), 1114–1155.

[101] Wegener, F., and Moradi, A. A first-order SCA resistant AES without fresh randomness. In *COSADE 2018: 9th International Workshop on Constructive Side-Channel Analysis and Secure Design* (Singapore, Apr. 23–24, 2018), J. Fan and B. Gierlichs, Eds., vol. 10815 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 245–262.

Curriculum Vitae

Siemen Dhooghe obtained his master's degree in mathematical engineering from KU Leuven in July 2017. In October 2017, he joined the COSIC research group at KU Leuven, in pursuit of a PhD degree. His research was focused on side-channel analysis, fault analysis, and symmetric key cryptography. Siemen has published in several conferences including Asiacrypt, FSE, CT-RSA, and CARDIS. During his PhD, Siemen taught exercise sessions for the course "Toegepaste Algebra" and supervised several thesis students.

List of Publications

1. Siemen Dhooghe and Svetla Nikova and Vincent Rijmen, "Threshold Implementations in the Robust Probing Model", TIS@CCS, 30–37, ACM, 2019

2. A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec, "Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols", IACR Transactions on Symmetric Cryptology 2020(3), pp. 1-45, 202

3. T. Beyne, S. Dhooghe, and Z. Zhang, "Cryptanalysis of Masked Ciphers: A not so Random Idea", In Advances in Cryptology - ASIACRYPT 2020, Lecture Notes in Computer Science, Springer-Verlag, 35 pages, 2020

4. Siemen Dhooghe and Svetla Nikova, "My Gadget Just Cares for Me - How NINA Can Prove Security Against Combined Attacks", CT-RSA, Lecture Notes in Computer Science, 12006, 35—55, Springer, 2020

5. Siemen Dhooghe and Svetla Nikova, "Let's Tessellate: Tiling for Security Against Advanced Probe and Fault Adversaries", CARDIS, Lecture Notes in Computer Science, 12609, 181—195, Springer, 2020

6. Siemen Dhooghe and Svetla Nikova, "Resilient Uniformity: Applying Resiliency in Masking," Cryptography and Communications 13(4), 18 pages, Springer, 2021

7. Tim Beyne, Siemen Dhooghe, Adrián Ranea, and Danilo Šijačić, "A Low-Randomness Second-Order Masked AES", Conference on Selected Areas in Cryptography (SAC), 2021

8. T. Beyne, S. Dhooghe, A. Moradi, A. R. Shahmirzadi, “Cryptanalysis of Efficient Masked Ciphers: Applications to Low Latency”, Accepted to TCHES 2022/1

9. S. Dhooghe, “Analyzing Masked Ciphers Against Transition and Coupling Effects”, Accepted to Indocrypt 2021



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
COSIC
Kasteelpark Arenberg 10, bus 2452
B-3001 Leuven
siemen.dhooghe@esat.kuleuven.be