TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Sicherheit in der Informationstechnik
an der Fakultät für Elektrotechnik und Informationstechnik

Algorithmic and Protocol Level Countermeasures to Protect Cryptographic Devices

Fabrizio De Santis

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		apl. Prof. Dr.-Ing. Helmut Gräb
Prüfer der Dissertation:	1.	Prof. Dr.-Ing. Georg Sigl
	2.	Prof. Dr.techn. Stefan Mangard

Die Dissertation wurde am 02.10.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 22.01.2018 angenommen.

Abstract

Nowadays the security of embedded devices does not only depend on the security of cryptographic algorithms, but also, and mainly, on the physical security of their implementations.

In fact, while the majority of cryptographic algorithms used today are founded on solid principles and underwent years of cryptanalysis, their implementations can often be broken by powerful physical attacks, such as side-channel and fault attacks.

The main challenge in the field of embedded cryptographic devices is to develop cryptographic implementations which are resistant against both side-channel and fault attacks, while retaining acceptable security levels within budget and physical constraints, *e.g.,* silicon area, execution time, power consumption, and energy.

Although many side-channel and fault countermeasures have been proposed over the last two decades, no conclusive nor satisfactory "costs vs. performance vs. security" trade-off has been found yet, *e.g.,* countermeasures are inefficient in terms of speed/area, they rarely meet market requirements, and are typically mutually contradictory.

This thesis provides a step forward towards designing and evaluating the security of state-of-the-art algorithmic-level and protocol-level countermeasures, which offer combined resistance against both side-channel and fault attacks.

As a first main contribution, we propose design principles, namely the selection of implementation-efficient addition chains, and methods, namely tower fields, to reduce the area size of Polynomial Masking Schemes (PMSs), *e.g.,* we reduce the area size of a polynomially masked Advanced Encryption Standard (AES) S-box to about that of a polynomially masked lightweight block-cipher like PRESENT.

In addition, we propose a shuffling scheme which inherently applies to polynomial masking schemes and increases both their side-channel and fault security.

We show that both our 1st-order implementation with shuffling and 2nd-order implementation are secure in a worst-case scenario analysis using Electro-Magnetic (EM) field measurements.

As a second main contribution, we show that the super-exponential security of Leakage-Resilient Pseudo-Random Functions (LRPRFs) can be reduced down to critical levels with the aid of EM micro-probes and multivariate template-based attacks.

In order to cope with such advanced high-resolution EM attacks, we propose a countermeasure based on spatial randomization and permutation networks to randomize the execution order of S-boxes along many instances.

Keywords: Polynomial Masking, Multi-Party Computation, Addition Chains, AES, Tower Fields, Shuffling, Leakage-Resilient PRF, Re-keying, Localized EM Attacks, Spatial Randomization, Permutation Networks.

Kurzfassung

Heutzutage hängt die Sicherheit von kryptographischen Geräten nicht nur von der Stärke der kryptographischen Algorithmen ab, sondern auch von der physischen Sicherheit ihrer Implementierungen.

Während die meisten kryptographischen Algorithmen, die heute verwendet werden, auf soliden mathematischen Konstruktionsprinzipien beruhen und jahrelanger Kryptoanalyse unterzogen wurden, können ihre Implementierungen mächtigen Angriffen, wie Seitenkanal- und Fehlerangriffen, oft nicht standhalten.

Die größte Herausforderung auf dem Gebiet der eingebetteten kryptographischen Geräte besteht darin, kryptographische Implementierungen zu entwickeln, die sowohl gegen Seitenkanal- als auch Fehlerangriffe resistent sind. Gleichzeitig soll ein akzeptables Sicherheitsniveau innerhalb eines Budgets und der technischen Einschränkungen ermöglicht werden.

Obwohl in den letzten zwei Jahrzehnten viele Seitenkanal- und Fehlergegenmaßnahmen in der Literatur vorgeschlagen wurden, ist für viele Anwendungen noch kein zufriedenstellender Kompromiss zwischen Wirtschaftlichkeit und Sicherheit gefunden worden. Die Gegenmaßnahmen sind oft ineffizient bezüglich Geschwindigkeit oder Fläche, sie erfüllen selten Marktanforderungen und verfolgen in der Regel widersprüchliche Ziele.

Diese Doktorarbeit ist ein Fortschritt zur Gestaltung und Bewertung der Sicherheit von hochmodernen algorithmischen und protokollorientierten Gegenmaßnahmen, die eine kombinierte Resistenz gegen Seitenkanal- und Fehlerangriffe bieten.

Als ersten Beitrag schlagen wir Designprinzipien vor, die auf der Auswahl von effizient zu implementierenden Additionsketten beruhen und Methoden, wie Tower-Fields, um die Fläche von Polynomial Masking Schemes (PMSs) zu reduzieren. Wir können dadurch die Fläche einer polynomiel maskierten Advanced Encryption Standard (AES) S-Box auf die Fläche von circa einer polynomisch maskierten leichtgewichtigen Block-Chiffre wie PRESENT reduzieren.

Darüber hinaus schlagen wir ein Verschachtelungsschema vor, das in die polynomielle Maskierung integriert werden kann und sowohl die Seitenkanal- als auch die Fehlersicherheit erhöht.

Wir zeigen, dass sowohl unsere Implementierung erster Ordnung mit Shuffling als auch unsere Implementierung zweiter Ordnung in einer Worst-Case-Szenarioanalyse mit Electro-Magnetic (EM) Messungen sicher sind.

Als zweiten Beitrag zeigen wir erstmals, dass die super-exponentielle Sicherheit von Leakage-Resilient Pseudo-Random Functions (LRPRFs) durch hochauflösende elektromagnetische (EM) Angriffe, mit dem Einsatz von Mikro-Sonden und multivariaten Template-basierten Angriffen, auf kritische Werte reduziert werden kann.

Um solche hochauflösenden EM-Angriffen zu bewältigen, schlagen wir als Gegenmaßnahme die räumliche Randomisierung über viele S-box Instanzen vor.

Schlüsselworte: Polynomial Masking, Multi-Party Computation, Addition Chains, AES, Tower Fields, Shuffling, Leakage-Resilient PRF, Re-keying, Localized EM Attacks, Spatial Randomization, Permutation Networks.

Acknowledgements

I thank Univ.-Prof. Dr.-Ing. Georg Sigl for giving me the chance to pursue a Ph.D. degree at the Technische Universität München (TUM), Univ.-Prof. Dipl.-Ing. Dr.techn. Stefan Mangard for providing scientific guidance and highly valuable advices, and the Chipcard&Security Division of Infineon AG for partially financing my research.

I thank Prof. François-Xavier Standaert for introducing me to the topic of Leakage-Resilient Pseudo-Random Functions (LRPRFs) and supervising my first steps in this field. Further, I am thankful to Dr.-Ing. Johann Heyszl and Dr. Lubos Gaspar for the close collaborations and profitable discussions.

I thank Univ.-Prof. Dipl.-Ing. Dr.techn. Stefan Mangard, Dr. Berndt Gammel, and Dr. Wieland Fischer for the opportunity to organize the laboratory course of the lecture "Secure Implementation of Cryptographic Devices" at TUM.

I thank my colleagues at TUM and Fraunhofer AISEC, Prof. Jens-Peter Kaps, Dr. Stefan Rass, and Dr. Marc Stöttinger for all the good times we shared together and the productive collaborations: Johanna Baehr, Mathieu Gross, Oscar Guillen, Matthias Hiller, Nisha Jakub, Philipp Koppermann, Hermann Seuscheck, Robert Specht, Lars Tabelmann, Michael Tempelmeier, Florian Unterstein, Markus Wamser, and Florian Wilde.

I express my gratitude to Marion Zillner, Harry Olm, Dr.-Ing. Martin Strasser, and Dr.-Ing. Michael Pehl for providing daily support in all kind of organizational and technical issues.

I am thankful to some very talented and hard-working students who supported my research and teaching activities in various ways: Tobias Bauer, Matthias Fischer, Omar Grati, Michael Gruber, Matthias Kneidel, Patrick Kresmer, Ermin Sakic, Arun Sattanathan, Andreas Schauer, Silvan Streit, Stefan Theil, and Thomas Zeschg.

Finally, a special thank goes to my family and friends for their love, patience, and encouragement over these years.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
ASIC	Application-Specific Integrated Circuit
BGA	Ball Grid Array
CBC-MAC	Cipher Block Chaining Message Authentication Code
CFB	Cipher FeedBack
CHES	Cryptographic Hardware and Embedded Systems
CMOS	Complementary Metal Oxide Semiconductor
CMS	Consolidated Masking Scheme
CPA	Correlation Power Analysis
DC	Direct Current
DES	Data Encryption Standard
DFA	Differential Fault Analysis
DFF	D-type Flip-Flop
DOM	Domain Oriented Masking
DPA	Differential Power Analysis
DRM	Digital Rights Management
EM	Electro-Magnetic
FA	Fault Attack
FCOFA	Faulty Ciphertext-Only Fault Attack
FIB	Focused Ion Beam
FIPS	Federal Information Processing Standard
FPGA	Field Programmable Gate Array
FYD	Fisher-Yates-Durstenfeld
GE	Gate Equivalent
GGM	Goldreich-Goldwasser-Micali
HOTI	Higher-Order Threshold Implementation
IC	Integrated Circuit
LDO	Low-Dropout Regulator
LFSR	Linear Feedback Shift Register

LRPRF	Leakage-Resilient Pseudo-Random Function
LUT	Look-Up Table
MIA	Mutual Information Analysis
MOSFET	Metal-Oxide-Semiconductor Field Effect Transistor
MPC	Multi-Party Computation
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NVM	Non-Volatile Memory
OFB	Output FeedBack
PCA	Principal Component Analysis
PCATA	PCA-based Template Attack
PCB	Printed Circuit Board
PMS	Polynomial Masking Scheme
POI	Points of Interest
PRF	Pseudo-Random Function
PRG	Pseudo-Random Generator
PRNG	Pseudo Random Number Generator
PRP	Pseudo-Random Permutation
RDI	Random Delay Insertion
RFID	Radio-Frequency IDentification
RP	Random Permutation
RPI	Random Process Interrupt
RSI	Random Start Index
SABL	Sense Amplifier Based Logic
SCA	Side-Channel Attack
SHA-3	Secure Hash Algorithm 3
SNR	Signal-to-Noise Ratio
SPA	Simple Power Analysis
SPN	Substitution-Permutation Network
SR	Success Rate
SSSS	Shamir's Secret Sharing Scheme
SVD	Singular Value Decomposition
TA	Template Attack
TI	Threshold Implementation
TOR	The Onion Router
TTP	Trusted Third Party
USB	Universal Serial Bus
WDDL	Wave Dynamic Differential Logic

1 Introduction

Contents

Since ancient times cryptographic methods were used to hide information and prevent secret disclosures. However, they mostly remained confined within military, diplomatic, and intelligence circles until the beginning of the twentieth century. Only with the advent of computers and advances in computer science and engineering, they finally stepped into the industry and scientific community, finding many industrial and civil applications, *e.g.*, to protect banking transactions, to secure online payments, to provide support for digital identification like passports and identification cards, to track commercial products using Radio-Frequency IDentification (RFID) systems, to maintain safe operation of medical and automotive products, and to provide online privacy mechanisms such as The Onion Router (TOR).

In our present society, cryptography is ubiquitous and transparently used by millions of users, *e.g.*, when opening electronic car doors, when performing payments with credit cards, or when using messaging applications on mobile phones. Indeed, the number of interconnected embedded devices, which rely on secure elements to provide cryptographic services like confidentiality, integrity, and authentication, is steeply increasing: according to [46], "*the number of secure elements shipped in* 2017 *is to exceed* 10 *billion*".

In order to prevent large-scale disasters, it is desirable to build cryptographic system that can resist any known attacks, but also prevent possible future threats, *e.g.*, the U.S. National Institute of Standards and Technology (NIST) recently reported *"that within the next 20 or so years, sufficiently large quantum computers will be built to break essentially all public key schemes currently in use ... [while] for symmetric key systems, one simple heuristic is to double the key lengths to compensate for the quadratic speedup achieved by Grover's algorithms ... [despite] this recommendation, does not take into account the possibility of more sophisticated quantum attacks"* [27].

Today, one of the most serious threats to the security of embedded cryptographic systems is given by implementation attacks, such as side-channel and fault attacks.

Outline

- **Section 1.1** provides an introduction to secret-key cryptography and defines the adversary models which are most relevant today.
- **Section 1.2** presents the motivations which are behind this work.
- **Section 1.3** highlights the main contributions provided by this thesis.
- **Section 1.4** provides an outline about the remainder of this thesis and concludes the chapter.

1.1 Secret-Key Cryptography

Cryptography aims at providing a set of algorithms to achieve different security goals [77, Sec. 1.2].

The most fundamental security goals are confidentiality, authentication, and integrity. *Confidentiality* means that information is kept secret to all unintended recipients, *integrity* means that all transmitted data can not be altered maliciously, and *authentication* means that the communication entities are really those who they claim to be.

Cryptographic algorithms can be broadly classified into two categories: secret-key and public-key algorithms.

Secret-key algorithms take a message and a secret key as input to produce an output which looks random to any polynomially bounded adversary. When a group of entities wants to communicate using secret-key cryptography, they have to share a secret key beforehand using a *secure channel*. This type of algorithms are also commonly referred to as *symmetric algorithms*.

Public-key algorithms use an additional key as input, which is public. In this case, when a group of entities wants to communicate using public-key cryptography, then each entity can transmit its own public key over an *insecure channel* and keep its own secret key undisclosed. This type of algorithms are also commonly referred to as *asymmetric algorithms*. Differently from secret-key algorithms, public-key algorithms do not require a pre-shared secret key, but they require a Trusted Third Party (TTP) to guarantee for the identities of the communicating parties.

In this work, only *secret-key encryption* algorithms based on *block-ciphers* are considered, and the term "key" will be often used as a synonym for "secret key".

A block-cipher can be seen as a family $\mathcal{P}$ of pseudo-random permutations BC_k indexed by a secret key $k \xleftarrow{\$} \{0,1\}^\kappa$ chosen at random from the set $\mathcal{U}_\kappa = \{0,1\}^\kappa$ of all binary strings of length κ, where κ is the security parameter and n is the block size:

$$\mathcal{P} = \{\mathsf{BC}_k : \{0,1\}^n \to \{0,1\}^n\}.$$

In practice, the majority of block-ciphers are product ciphers which iterate a round function. The round function applies different transformations of the input according to the properties of *confusion* and *diffusion* [112]. Confusion means that each output bit depends on several bits of the key and structured relationships between the input and output bits are avoided. Diffusion means that if a single bit of the input changes, then about half of the bits in the output change, and vice versa.

Note that block-cipher encryption algorithms are typically sufficient to achieve the fundamental security goals of confidentiality, integrity, and authentication. For instance, confidentiality can be achieved using either one between Cipher FeedBack (CFB) and Output FeedBack (OFB) mode of operations, integrity can be obtained using the Davies-Meyer construction, and authentication can be realized using Cipher Block Chaining Message Authentication Code (CBC-MAC).

More information on block-cipher mode of operations can be found in [87, Sec. 5].

1.1.1 Adversary Models

In order to specify the security guarantees of a cryptographic system, a definition of the capabilities of the adversary and what constitutes a successful attack should be defined.

An adversary model describes “which kind” and “what amount” of information is made accessible to the adversary when analyzing the security of a cryptographic system.

A cryptosystem is typically considered as “broken” when the adversary can either (generically) distinguish the output of a cryptographic algorithm from a truly random string or (more concretely) recover the secret key by exploiting the information which is available from the model.

Today, the most relevant adversary models are the black-box model, the gray-box model, and the white-box model.

Black-box Model In this model, cryptographic implementations are considered as black-boxes, which the adversary can (adaptively) query by providing a certain limited number of inputs, *e.g.*, the adversary can feed a block-cipher with plaintexts and obtain the corresponding ciphertexts generated using a certain secret key.

The adversary does not have access to either the implementation or its execution environment, but all details about the internal machinery of the underlying algorithm are considered as known. This model is realistic only if the adversary has no access (either physical or remote) to the platform implementing the cryptographic algorithm, *i.e.*, the adversary can only query the algorithm itself, but obtains no additional information from its physical realization.

Two of the most prominent types of attacks in the black-box model are differential cryptanalysis attacks [13] and linear cryptanalysis attacks [73].

Gray-box Model The gray-box model is an extension of the black-box model, where the adversary has additional access to the physical implementation of a cryptographic system and can observe its physical properties, like the power consumption or Electro-Magnetic (EM) field variations, or may even tamper with it, *e.g.*, the adversary can inject faults into the device in order to obtain information about the secret key.

Today’s embedded cryptographic devices, such as RFID or smart cards, should provide security guarantees in the gray-box model (at least). In fact, while the security margins of modern cryptographic primitives are currently considered adequate in the black-box model (for most applications), the security of the corresponding cryptographic implementations in the gray-box model is still a vivid area of research.

The most prominent types of attacks in the gray-box model are Differential Power Analysis (DPA) attacks [71, Ch. 6] and Differential Fault Analysis (DFA) attacks [14].

White-box Model In this model, the adversary has full control over a cryptographic system and its environment, *i.e.*, the adversary can observe and modify intermediate data of computations as well as manipulate the execution of cryptographic code with the aid of debuggers and breakpoints, or even search the memory for secret keys [29].

This model finds its main application in the field of Digital Rights Management (DRM) and mainly affects software implementations. Today, the white-box model represents the worst-case scenario and can be seen as an extension of the gray-box model towards yet more powerful adversaries.

Several practical attacks have been already developed against implementations of Data Encryption Standard (DES) and Advanced Encryption Standard (AES) in the white-box model [18, 127]. Very recently, also implementation attacks such as DPA attacks were successfully performed against white-box implementations [104, 20].

1.1.2 Security Definitions

The security of cryptographic systems is typically expressed in terms of time and space complexities required to perform a successful attack as a function of a security parameter, *e.g.*, the key length κ.

The security level of a cryptographic system is always upper bounded by the size of the key space 2^κ, *i.e.*, the key space has to be large enough in order to prevent exhaustive key enumerations. The lower bound is represented by the best known attack. In general, it holds that the security level of any cryptographic systems is a monotone decreasing function of time.

According to the Kerckhoffs' principle [64], the security of a cryptographic algorithm should only depend on the secrecy of its key. In other words, a cryptographic system should remain secure, even if its algorithmic operations are known to the adversary.

In light of this principle, several modern cryptographic standards, like the Advanced Encryption Standard (AES) [85] and Secure Hash Algorithm 3 (SHA-3) [86], underwent a public scrutiny and received an extensive analysis from a large number of researchers and experts.

If an adversary can distinguish the output of a block-cipher from a random string with a significant advantage and less effort than exhaustive key search, then the cipher is considered as broken, even if such a break does not immediately lead to any security failure in practice.

A block-cipher is perfectly secure in the black-box model, if the knowledge of the ciphertext c does not lead to any information about the plaintext p:

$$\Pr\left[P = p \mid C = c\right] = \Pr\left[P = p\right].$$

A block-cipher is perfectly secure in the gray-box model, if the knowledge of the ciphertext c and the side-channel information ℓ do not lead to any information about the plaintext p:

$$\Pr\left[P = p \mid C = c, L = \ell\right] = \Pr\left[P = p\right].$$

In practice, the security of cryptographic implementations in the gray-box model is typically assessed using heuristic security metrics such as the Guessing Entropy (GE) or Success Rate (SR) [116].

The GE is defined as the average number of key candidates that have to be tested after a side-channel attack in order to recover the secret key, while the SR is defined as the number of measurements which are required to achieve a certain success probability of recovering the secret key after side-channel attacks.

Yet another and more effective method to evaluate the security of cryptographic implementations after side-channel attacks is given by key rank estimation algorithms [121, 93]. Key rank estimation algorithms enable designers to approximate the security level of cryptographic implementations by approximating the position of the secret key in the list of all possible key candidates after side-channel attacks.

In remaining of this thesis, the term “security level” should be intended as “security level left after side-channel attacks”.

1.2 Motivation of This Work

With the increasing adoption of embedded devices which can easily fall into adversarial hands, like identification, banking, or transportation cards, the security of cryptographic devices does not only depend on the strengths of cryptographic algorithms (black-box model), but also, and mainly, on the physical security of their implementations (gray-box model).

The typical scenario where cryptographic embedded devices are employed is the following: two entities like an RFID tag and an RFID terminal want to securely communicate over an insecure channel relying on secret-key cryptography, *i.e.*, each entity stores the same secret key.

The RFID terminal is typically sealed, not easy accessible for the adversary, and has more computational power than the RFID tag. In contrast, the RFID tag is typically resource-constrained and can be easily obtained from the adversary, *i.e.*, the adversary can own a Pay-TV card for the time necessary to mount a physical attack on it, recover its secret key, and finally clone it.

In the last two decades, several side-channel analysis techniques have been developed to recover the secret keys stored into cryptographic devices. In parallel, a large body of countermeasures has been proposed to thwart side-channel attacks at different levels of abstraction, *e.g.*, at the physical level [96, 6], at the logic level [91, 79], at the algorithmic level [24, 82, 95], and at the protocol level [75, 10, 41].

Common to any such practices is the attempt to either hide, randomize, or bound the side-channel information leaked by cryptographic devices.

Typical countermeasures to thwart side-channel and fault attacks include secure logic styles, shielding techniques, noise generators to reduce the Signal-to-Noise Ratio (SNR), secret sharing schemes to conceal intermediate cryptographic computation steps, and re-keying schemes to bound the information leakage.

However, no conclusive "costs vs. performance vs. security" trade-off has been identified yet, *e.g.*, countermeasures lead to high performance degradation, they rarely meet time-to-market constraints, and are typically opposite to each other. In fact, while countermeasures against fault attacks usually require some type of redundancy, this latter typically amplifies the side-channel information leakage.

This thesis represents a step forward in the field of algorithmic and protocol level countermeasures with improved resistance against both side-channel and fault attacks.

1.3 Main Contributions and Open Issues

This thesis work describes advancements in the design and analysis of algorithmic-level masking and re-keying schemes aimed at providing higher side-channel and fault security along with improved performance and area-size.

In particular, the field of Polynomial Masking Scheme (PMS) and fresh re-keying schemes with Leakage-Resilient Pseudo-Random Functions (LRPRFs) are advanced.

1.3.1 Polynomial Masking Schemes

Masking schemes aim at making the leakage information of key-dependent intermediate computation steps look random to the adversary.

Polynomial masking was introduced at the workshop on Cryptographic Hardware and Embedded Systems (CHES) in 2011 and represents the first attempt to conceive both a glitch-resistant and higher-order masking scheme in literature. It is based upon Shamir's Secret Sharing Scheme (SSSS) and secure Multi-Party Computation (MPC) protocols [95].

A first implementation of the Advanced Encryption Standard (AES) on Field Programmable Gate Array (FPGA) was presented at CHES in 2013 [80], while a 1st-order and a 2nd-order polynomially masked implementation of the PRESENT S-box on FPGA were published in 2015 [36].

Polynomial masking offers some very appealing and unique features like:

1. it is glitch-resistant and higher-order by design;
2. it can be implemented with very regular design patterns, *i.e.*, it allows for an easy integration with secure logic styles;
3. it can be easily scaled to any order, *e.g.*, simple loops can be used in software or "VHDL Generics" can be used in hardware to specify the desired masking order: this feature allows to quickly adjust the masking order to pass a required security certification level;
4. it can be implemented once and re-used for different cryptographic algorithms after the principle "secure it once, re-use it for all";
5. it is "*intrinsically immune against fault attacks when less than one-third of the shares are corrupted*" [95, Sec. 6];
6. it can be inherently shuffled without performance loss, as it will be shown in Chapter 3.

However, PMSs suffer from two main drawbacks: they have poor performance figures in terms of speed, size, and randomness requirements when compared to other masking schemes such as Threshold Implementations (TIs) and 2nd-order univariate leakages have been shown to occur in the 1st-order scheme.

In this thesis work, the following contributions to improve the state of the art on PMSs are provided:

1. We introduce new principles for the selection of more efficient addition chains that lead to both faster and more compact implementations of cryptographic S-boxes. When applied to the AES S-box, our 1st-order implementation is 24% smaller, need 21% less clock cycles, and 6% less randomness than the state-of-the-art implementation presented at CHES in 2013 [80].

2. We show how tower fields perfectly adapt to PMSs due to their algebraic structure and can be used to significantly reduce the area and randomness requirements of higher-order polynomially masked implementation of S-boxes. Our proposed 1st-order tower-field AES S-box requires only ≈ 4 kGE in Application-Specific Integrated Circuit (ASIC), which is approximately the size of a 1st-order polynomially masked PRESENT S-box [36].

3. We propose a simple and effective shuffling-based countermeasure with the goal of *hiding* the univariate leakage of the 1st-order scheme at almost no area expenses and no time penalties. To the best of our knowledge, this is the first time that a masking scheme itself is shuffled during non-linear operations, *i.e.*, the shuffling scheme operates on the shares themselves leading to a shuffled masking scheme. Our scheme can be possibly combined with further shuffling techniques operating at higher levels of abstraction, *e.g.*, the order of the S-box look-ups can be additionally shuffled. Note our proposed countermeasure inherently increases also the resistance against fault attacks, since the fault injection probability is decreased by the timing disarrangements introduced by the shuffling module.

4. We provide a description of the hardware design for any masking order, synthesis results in ASIC up to the 6th-order, and perform a higher-order side-channel analysis showing that univariate security up to 10 million measurements can be achieved by both a 2nd-order implementation and a 1st-order implementation with shuffling.

1.3.2 Fresh Re-Keying with Leakage-Resilient PRFs

Leakage-resilient cryptography aims at developing cryptographic algorithms for which the physical security against side-channel attacks can be formally analyzed. Following the seminal work of [45], several symmetric cryptographic primitives have been formalized in this setting. Most of them, like leakage-resilient Pseudo-Random Generators (PRGs) and Pseudo-Random Functions (PRFs), can be instantiated with a block-cipher as underlying component.

The work of Medwed et al. at CHES 2012 proposed a Leakage-Resilient Pseudo-

Random Function (LRPRF) based on AES and showed that AES may not be the best block-cipher for integration in a LRPRF. Also, it left a number of open questions regarding the security of the new proposal and which block-ciphers are better suited for LRPRFs [76].

These questions were later on addressed in [10], where the security of a block-cipher based LRPRF was studied at different abstraction levels. In particular, the work of [10] analyzes possible block-cipher designs and attacks exploiting specific features of the algorithmic description, hardware architecture, and physical implementation of LRPRFs.

In this thesis work, the following contributions to improve the state of the art on LRPRFs are provided:

1. We study the impact of high-resolution EM analysis on the security of a LR-PRF implementation on FPGA by showing that the resulting (key-dependent) algorithmic noise remains difficult to exploit in the univariate setting when considering both time-based and location-based leakage information.
2. We extend the previous analysis to the multivariate setting by showing for the first time that the super-exponential security of LRPRFs can indeed be reduced down to critical levels using multivariate high-resolution EM analysis.
3. We propose a spatial randomization countermeasure based on permutation networks aimed at disarranging the execution of parallel S-boxes over different circuitries to thwart high-resolution EM analysis. This contribution is not specific and limited to LRPRFs only, but it can be straightforwardly applied to any SPN-based block-ciphers with a bit-permutation layer.

1.4 Thesis Outline

Chapter 2 provides background information on power and EM measurements, implementation attacks, and countermeasures. The remaining chapters of the thesis are conceptually divided into two parts.

The first part is dedicated to polynomial masking schemes and consists of the following chapters:

- **Chapter 3** introduces polynomial masking, provides new principles for the selection of hardware-efficient addition chains, describes the hardware design of a higher-order polynomially masked AES S-box using the addition chains selected according to the previous principles, and details on how to embed shuffling into polynomial masking schemes.

- **Chapter 4** shows how tower field arithmetic can be used in combination with polynomial masking to drastically reduce the area size of a polynomially masked AES S-box.
- **Chapter 5** presents side-channel analysis results of the previously described implementations in the EM domain showing that univariate security can be achieved by both a 2$^{\text{nd}}$-order implementation and a 1$^{\text{st}}$-order implementation with shuffling.

The second part is dedicated to Leakage-Resilient Pseudo-Random Functions (LRPRFs) and consists of the following chapters:

- **Chapter 6** introduces leakage-resilient PRFs and shows how high-resolution EM template-based attacks can be used to defeat the super-exponential security of LRPRF implementations on FPGAs.
- **Chapter 7** proposes a shuffling countermeasure to spatially randomize leakage-resilient PRFs based on permutation networks.

Chapter 8 concludes the thesis with a brief summary of the main contributions and an outlook on future challenges.

2 Background

Contents

Nowadays, state-of-the-art cryptographic standards for secret-key cryptography, like the Advanced Encryption Standard (AES) and Secure Hash Algorithm 3 (SHA-3), are widely believed to have sufficient security margins in the black-box model.

Indeed, only a breakthrough in current cryptanalytic techniques may affect the security of current secret-key algorithms in this model. Nevertheless, it is already advisable to double the key size of symmetric algorithms in order to ensure long-term security and prevent secrets to be revealed in the future, when large quantum computers could be built [5].

The major threat to the security of today's cryptographic systems are implementation attacks in the gray-box model, such as side-channel and fault attacks. This type of attacks allows to recover the secret key material from the information leaked by the physical characteristics of a cryptographic implementation, such as execution time and power consumption.

Implementation attacks are amongst the strongest types of attacks against embedded cryptographic devices for costs and effectiveness. They can be applied in all such cases where the adversary can measure or manipulate physical characteristics of a cryptographic implementation, like in the case of smart cards, Universal Serial Bus (USB) tokens, Radio-Frequency IDentification (RFID) tags, and similar devices.

While no specific knowledge about the internals of cryptographic implementations is typically required, the adversary usually needs to own the device for the time necessary to setup a successful attack.

Outline

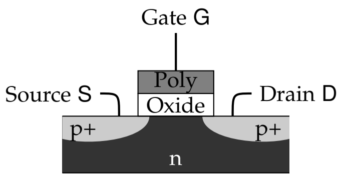
- **Section 2.1** provides an overview on power consumption in Complementary Metal Oxide Semiconductor (CMOS) devices and power measurements of Integrated Circuits (ICs).
- **Section 2.2** provides an overview on EM fields, EM measurements on ICs, IC package decapsulation, and EM probes.
- **Section 2.3** provides a classification of implementation attacks with a focus on side-channel and fault attacks.
- **Section 2.4** provides an overview on implementation countermeasures like masking, hiding, and re-keying schemes.

2.1 Power Consumption in CMOS Devices

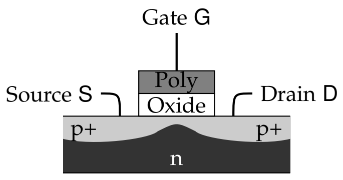
The building blocks of ICs are logic cells which realize elementary logic functions, such as AND, OR, and XOR functions.

Nowadays, logic cells are typically fabricated in CMOS technology, which uses complementary pairs of P-type and N-type Metal-Oxide-Semiconductor Field Effect Transistors (MOSFETs) - abbreviated as pMOS and nMOS, respectively.

Transistors can be used as simple switches that can be turned either “ON” or “OFF”. MOSFET transistors are fabricated using either negatively doped or positively doped



(a) If $V_{\mathsf{GD}} \geq V_{\mathsf{T}}$, then the two regions $p+$ are isolated and the pMOS is turned "OFF".



(b) If $V_{\mathsf{GD}} < V_{\mathsf{T}}$, then a channel is created between the two regions $p+$ and the pMOS is turned "ON".

Figure 2.1: Cross-section of a pMOS Transistor consisting of a substrate of negatively doped silicon (n) separating two areas of positively doped silicon ($p+$), capped with an insulator material (Oxide) and conducting material (Poly).

silicon. Negatively doped silicon is rich in electrons, while positively doped silicon is rich in holes.

Figure 2.1 shows the cross-section of a pMOS transistor consisting of a substrate of negatively doped silicon (n) which separates two areas of positively doped silicon ($p+$). The area separating the two $p+$ regions is capped with an insulator material (Oxide) and conducting material (Poly).

MOSFET transistors have three contacts: a gate G, a source S, and a drain D. The voltage drop V_{GD} between the gate and drain controls the "ON/OFF" switching operation of transistors. In nMOS transistors current flows from S to D ("ON" State), if V_{GD} is greater than a certain threshold voltage V_{T}. In contrast, pMOS transistors are conducting ("ON" State) if $V_{\mathsf{GD}} < V_{\mathsf{T}}$.

Currents flowing in transistors dissipate energy as heat. Power is the energy dissipated by transistors per unit of time. The total power consumption P of a circuit $\mathcal{C}$ is the sum of the power dissipated by each logic cell c composing the circuit:

$$\mathsf{P} = \sum_{c \in \mathcal{C}} \mathsf{P}_c \tag{2.1}$$

The power P_c dissipated by a logic cell c consists of two components: *static power* $\mathsf{P}^{\mathsf{s}}_c$, which is the power dissipated when the cell is holding its logic state, and *dynamic power* $\mathsf{P}^{\mathsf{d}}_c$, which is the power dissipated when the cell output (or any of its internal parts) is switching from one logic state to another, *e.g.*, during an output transition from $0 \rightarrow 1$:

$$\mathsf{P}_c = \mathsf{P}^{\mathsf{s}}_c + \mathsf{P}^{\mathsf{d}}_c \tag{2.2}$$

Static power consumption $\mathsf{P}_c^{\mathsf{s}}$ is state, temperature, and voltage dependent, and is mainly due to reduced threshold voltages that prevent gates from completely turning off.

Dynamic power consumption is given by the sum of the *switching power* $\mathsf{P}_c^{\mathsf{sw}}$ and the *internal switching power* $\mathsf{P}_c^{\mathsf{intsw}}$, which are the powers dissipated due to the charging and discharging of the output and internal load capacitances, respectively, and the *short-circuit power* $\mathsf{P}_c^{\mathsf{sc}}$, which is the power dissipated due to paths that are temporarily established between V_{DD} and ground during logic transitions:

$$\mathsf{P}_c^{\mathsf{d}} = \mathsf{P}_c^{\mathsf{sw}} + \mathsf{P}_c^{\mathsf{intsw}} + \mathsf{P}_c^{\mathsf{sc}} \tag{2.3}$$

For a long time, dynamic power consumption has represented the major contribute to the total power consumption dissipated by CMOS integrated circuits, being static power consumption orders of magnitude smaller than dynamic power consumption, *i.e.*, the more cells changed their state, the more power was dissipated.

However, with the scaling of CMOS technology under 100 nm, static power consumption has started being a non negligible source of power dissipation which is expected to take over dynamic power consumption in the near future as technology size further scales down.

In order to get a deeper understanding about the root causes of power consumption in CMOS devices, a simple model of a CMOS inverter cell is considered (cf. Figure 2.2). The power consumption for more complex gates can be similarly derived.

Power Consumption in a CMOS Inverter Cell The static power consumption in a CMOS inverter cell inv is given by the product of the leakage current $I_\ell = I_{\ell(p)} + I_{\ell(n)}$ and the supply voltage V_{DD}:

$$\mathsf{P}_{\mathsf{inv}}^{\mathsf{s}} = V_{\mathsf{DD}} \cdot [I_{\ell(p)} + I_{\ell(n)}] = V_{\mathsf{DD}} \cdot I_\ell \tag{2.4}$$

The switching power is mainly due to the charging and discharging of the load capacitances at the output of the inverter cell C_{ld}, *i.e.*, the net and gate capacitances on the driving output:

$$\mathsf{P}_{\mathsf{inv}}^{\mathsf{sw}} = \alpha \cdot f_{\mathsf{clk}} \cdot C_{\mathsf{ld}} \cdot V_{\mathsf{DD}}^2, \tag{2.5}$$

where f_{clk} is the clock frequency and α is the activity factor of the cell, *i.e.*, the average number of transitions that occur at the inverter output in one clock cycle [71].

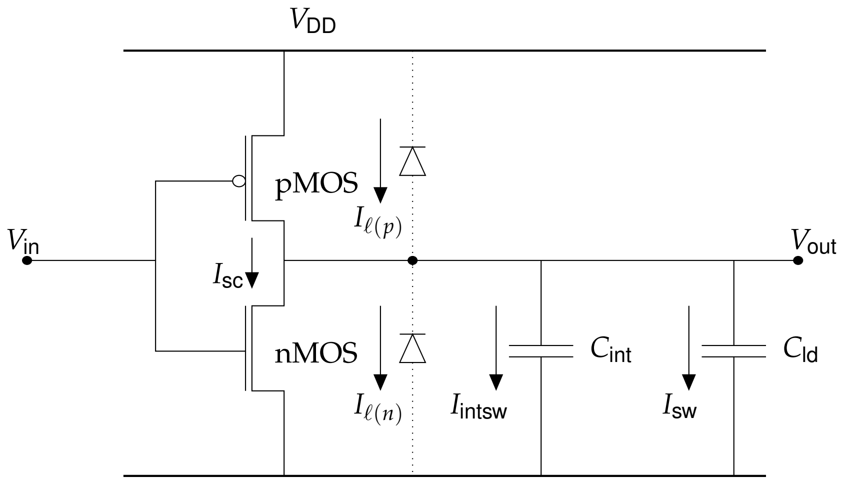


Figure 2.2: Model of power consumption in a CMOS inverter cell: it depends on currents flowing inside the cell (I_{sc}, I_{ℓ}, and I_{intsw}) and the current flowing to output load (I_{sw}).

Similarly, the internal switching power $\mathsf{P}_{\mathsf{inv}}^{\mathsf{intsw}}$ is dissipated due to charging and discharging of the internal load capacitance C_{int}:

$$\mathsf{P}_{\mathsf{inv}}^{\mathsf{intsw}} = \alpha \cdot f_{\mathsf{clk}} \cdot \mathsf{C}_{\mathsf{int}} \cdot V_{\mathsf{DD}}^2 \tag{2.6}$$

Short-circuit power $\mathsf{P}_{\mathsf{inv}}^{\mathsf{sc}}$ is dissipated when both pMOS and nMOS transistors are temporary conducting during state transitions of the inverter cell:

$$\mathsf{P}_{\mathsf{inv}}^{\mathsf{sc}} = \alpha \cdot f_{\mathsf{clk}} \cdot I_{\mathsf{sc}} \cdot \tau_{\mathsf{sc}} \cdot V_{\mathsf{DD}} = \alpha \cdot f_{\mathsf{clk}} \cdot \mathsf{C}_{\mathsf{sc}} \cdot V_{\mathsf{DD}}^2, \tag{2.7}$$

where τ_{sc} is the time when the short-circuit current I_{sc} is flowing, and $C_{\mathsf{sc}} = I_{\mathsf{sc}} \cdot \tau_{\mathsf{sc}} / V_{\mathsf{DD}}$. The total power consumption of a CMOS inverter cell is summarized in Table 2.1.

2.1.1 Power Measurements of ICs

Power measurements of digital integrated circuits can be taken with the aid of a digital oscilloscope coupled to either a current probe or a voltage probe. Voltage probes are more widely employed in the context of side-channel analysis due to their typically wider bandwidths (25 MHz - 500 MHz) and lower input capacitances ($<$ 10 pF).

The most convenient way to take power measurements of an IC is to insert a shunt resistor in the path between either a ground pin of the IC and the ground plate or the power supply V_{DD} and a power pin of the IC.

Table 2.1: Power consumption in a CMOS inverter cell consists of two parts: static power consumption $\mathsf{P}_c^{\mathsf{s}} = V_{\mathsf{DD}} \cdot I_\ell$ and dynamic power consumption $\mathsf{P}_c^{\mathsf{d}} = \alpha \cdot f_{\mathsf{clk}} \cdot [(C_{\mathsf{int}} + C_{\mathsf{ld}} + C_{\mathsf{sc}}) \cdot V_{\mathsf{DD}}^2]$. This latter contribution is present only if there is an input transition.

Input Transition	Output Transition	Power Consumption
$0 \to 0$	$0 \to 0$	$V_{\mathsf{DD}} \cdot I_\ell$
$0 \to 1$	$1 \to 0$	$V_{\mathsf{DD}} \cdot I_\ell + \alpha \cdot f_{\mathsf{clk}} \cdot [(C_{\mathsf{int}} + C_{\mathsf{ld}} + C_{\mathsf{sc}}) \cdot V_{\mathsf{DD}}^2]$
$1 \to 0$	$0 \to 1$	$V_{\mathsf{DD}} \cdot I_\ell + \alpha \cdot f_{\mathsf{clk}} \cdot [(C_{\mathsf{int}} + C_{\mathsf{ld}} + C_{\mathsf{sc}}) \cdot V_{\mathsf{DD}}^2]$
$1 \to 1$	$1 \to 1$	$V_{\mathsf{DD}} \cdot I_\ell$

These two possible configurations are illustrated in Figure 2.3 using a simplified ideal model which does not consider either multiple ground/power pins nor the presence of other circuital components.

The first configuration has the advantage that single-ended passive voltage probes can be used to measure the voltage drop over the resistor V_{R} (cf. Figure 2.3a). This type of probes is typically very cheap. As a drawback, the I/O pins of the IC could provide alternate ground paths in this configuration causing variable bias in the measurements.

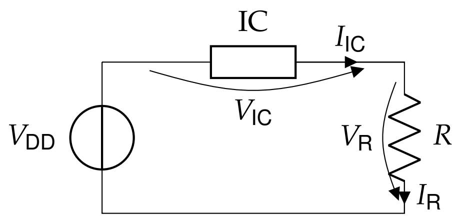
The second configuration does not have the drawbacks of the first one, *i.e.,* the current delivered by the voltage power supply V_{DD} to the chip can be directly measured and it does not present issues with multiple ground paths or loops (cf. Figure 2.3b).

However, differential active probes are needed to handle the common-mode voltage. As a drawback, this type of probes is typically more expensive, have lower bandwidths, and suffer from higher measurement noise due to the presence of active components.

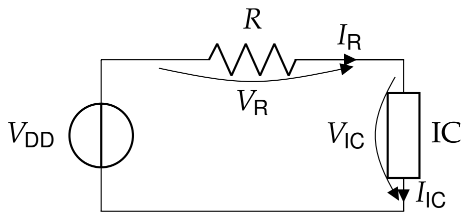
In both cases, the average power P_{IC} dissipated by the IC is related to the voltage drop over the resistor V_{R} as follows:

$$\mathsf{P}_{\mathsf{IC}} = V_{\mathsf{IC}} I_{\mathsf{IC}} = (V_{\mathsf{DD}} - V_{\mathsf{R}}) \frac{V_{\mathsf{R}}}{R},$$

where $I_{\mathsf{IC}} = I_{\mathsf{R}} = \frac{V_{\mathsf{R}}}{R}$ is the average current absorbed by the IC and R is the electrical resistance of the shunt resistor. In the context of side-channel analysis, the instant power consumption $\mathsf{P}_{\mathsf{IC}}(t)$ depends on the amount of current $I_{\mathsf{IC}}(t)$ that is instantaneously drawn from the IC during cryptographic computations. In practice the electrical resistance R of the shunt resistor should be chosen in respect to the IC under



(a) Shunt resistor between the IC and ground: measurements can be performed with passive single-ended voltage probes.



(b) Shunt resistor between the power supply and the IC: measurements can be performed with active differential voltage probes.

Figure 2.3: Power measurements using a shunt resistor: a shunt resistor is inserted in the circuit to measure the flow of electric currents in a IC by sensing the variable voltage drop across it.

test, *e.g.,* it should be “small enough” in order to do not affect the normal operation of the circuit, but also “large enough” in order to provide a good Signal-to-Noise Ratio (SNR). Typical values of R vary between $0.5\,\Omega$ and $50\,\Omega$ for conventional embedded devices.

In the context of side-channel analysis, power measurements have practically two main limitations, namely:

1. The available bandwidth is typically limited by different parasitics laying on the path between the measurement point and the Analog-to-Digital Converter (ADC) of the oscilloscope, *e.g.,* due to the bonding wires in the die’s package and decoupling capacitors on the Printed Circuit Board (PCB) board. In other words, power measurements are always low-passed to some extent, due to the RLC circuit that is established on the path to the ADC, thus making impractical to exploit the information contained in the higher-frequency components of the leakage spectrum.

2. Power measurements require to modify the PCB board in order to insert a shunt resistor in the circuit. This is not always an easy task, *e.g.,* when multi-layers PCB boards with multi-ground planes and several Low-Dropout Regulators (LDOs) in the power supply paths are used.

Due to these limitations, EM measurements are often employed for side-channel analysis in place of power measurements, being EM field variations directly related to the electrical activity of ICs. This latter ultimately depends on the cryptographic operations which are performed on the device.

2.2 Electro-Magnetic Fields

EM fields are composite vector fields which consist of electric fields and magnetic fields. While electric fields are produced by particles provided with electric charge, magnetic fields are produced by moving charges, *i.e.*, current flows.

The electric field **E** and magnetic field **B** are defined by the Lorentz Force Law, which describes the way a force **F** acts on a particle with electric charge q and instantaneous velocity **v** in presence of an electric field **E** and a magnetic field **B** as:

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}),$$

where $\times$ is the vector cross product operator.

The way how stationary and moving charges give rise to electric and magnetic fields is described by Maxwell's equations:

1. Gauss's law states that the electric flux through a closed surface is proportional to the enclosed charge:
$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0},$$
where $\nabla \cdot \mathbf{E}$ is the divergence of the electric field, ε_0 is the electric constant, and ρ is the total electric charge density.

2. Gauss's law for magnetism states that the total magnetic flux through a closed surface is zero, *i.e.*, there are no isolated magnetic monopoles:
$$\nabla \cdot \mathbf{B} = 0,$$
where $\nabla \cdot \mathbf{B}$ is the divergence of the magnetic field.

3. Maxwell-Faraday equation is a generalization of Faraday's law of induction which describes how time-varying magnetic fields generate electric fields:
$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

4. Ampère's law relates the magnetic field back to its origin stating that the magnetic field induced around a closed loop is proportional to the sum of the electric current and displacement current it encloses:

$$\nabla \times \mathbf{B} = \mu_0 \left(\mathbf{J} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right),$$

   where $\mathbf{J}$ is the current density and μ_0 is the magnetic constant.

In addition to $\mathbf{B}$, there is another quantity denoted by $\mathbf{H}$, which further accounts for how strong a region itself is magnetized. The two quantities are related by the following formula:

$$\mathbf{H} = \frac{\mathbf{B}}{\mu_o} - \mathbf{M},$$

where $\mathbf{M}$ is the magnetization field, and represents how strongly a region of material is magnetized. In vacuum, $\mathbf{M} = 0$, thus $\mathbf{B} = \mu_o \mathbf{H}$. Note that all fields are functions of the position in the three dimensional space $\mathbf{r}$ and time t, *e.g.*, $\mathbf{B} \equiv \mathbf{B}(\mathbf{r}, t)$.

In general, it holds that the higher is the charge, the stronger is the electric field at a given distance, and the higher is the current flow, the stronger is the magnetic field at a given distance.

Near and Far Fields Electromagnetic fields can be broadly divided into two types: near fields and far fields.

Let $\lambda = \frac{c}{f}$ be the wavelength of an EM field, where $c \approx 3 \times 10^8$ m/s is the speed of light in vacuum and f is the frequency of its emissions. Then, the region located at a distance $d \ll \lambda$ from the source is typically referred to as the *near-field*. In this region, the electric and magnetic fields are directly generated by charges and currents, respectively.

The region located at a distance $d \gg 2\lambda$ from the source is typically referred to as the *far-field*. In this region, the electric and magnetic fields are caused by the respective other field propagating into space. In the region between λ and 2λ, both near-field and far-field effects are present.

2.2.1 EM Measurements of ICs

Today's ICs consist of millions of transistors and wires, where electric currents flow from nodes at higher potential to nodes at lower potential following various paths through logic gates and metal layers.

These currents together with stationary charges generate EM fields in the three-dimensional space surrounding the chip. The superposition of such fields results in a very complex EM field which is difficult to model in practice, *i.e.,* the modeling of the EM field surrounding a chip is typically not feasible with currently available software tools, as there are too many EM sources which either amplify or cancel the effect of each other out.

At a first approximation, an integrated circuit can be viewed as a loop antenna radiating a *magnetic* field through its surface. In fact, while most materials shield electric fields to some extent, magnetic fields are typically *not* attenuated by most materials. This makes magnetic field measurements typically more suitable for side-channel analysis.

In general, EM measurements offer some advantages over power measurements like reduced invasiveness, improved ranges of bandwidth, and improved spatial localization. In fact, they do not require the insertion of a resistor in series to the chip, are not low-passed by RLC circuits on the PCB board, and allow for a higher spatial resolution that permits to isolate the activity of small parts of the circuit under consideration.

2.2.2 EM Probes

The magnetic field variations in the EM field surrounding the chip can be measured using magnetic inductive probes. They are small electrically balanced loop antennas built with an electric field shield and a small break in the loop to prevent shield currents from flowing around the loop.

The probe's output voltage, *i.e.,* the induced electromotive force $\mathcal{E}$, is proportional to the negative variation of the magnetic flux ϕ_B sensed by the probe through Faraday's law of induction:

$$\mathcal{E} = -N\frac{d\phi_B}{dt} = -N\frac{d}{dt}\left(\iint_{\Sigma(t)} \mathbf{B} \cdot d\mathbf{A}\right),$$

where N is the number of loops in the coil, $\mathbf{B}$ is the magnetic field, d$\mathbf{A}$ is an infinitesimal area of the concatenated surface $\Sigma(t)$ comprised within the coil. Note that the coil can be moved over time t, *i.e.,* the area $\Sigma(t)$ that is concatenated by the probe may change over time and thus the magnetic flux through it. Further note that the negative sign of $\mathcal{E}$ is determined by Lenz's law, *i.e.,* any current generated by a varying magnetic field in a coil produces a magnetic field that opposes the change in the magnetic field that has induced it.

When performing EM measurements the following probe's parameters can be considered:

- *Orientation*:
  The probe orientation should match the direction of the EM field produced by the interested source. The three components of the magnetic field can be measured by rotating the probe with an angle of 90 °C around the (x, y, z)-axes.

- *Resolution*:
  The probe resolution should match the area source involved in the EM emissions of interest. Typical values ranges from 25 mm to 1 mm. If the probe has high resolution, then only parts of the emissions are captured. In contrast, if the probe has low resolution, then it might include emissions from other components.

- *Sensitivity*:
  The probe sensitivity should be sufficiently high to fully exploit the input range of the oscilloscope. Sensitivity and resolution are typically related parameters, as increasing the loop size typically increases the sensitivity, but also reduces the resolution. Note that amplifiers can be used to increase SNR of small-sized probes.

- *Bandwidth*:
  The probe bandwidth is typically defined as the range of frequencies at which a sinusoidal input signal is attenuated up to 70.7% of its original amplitude, also known as the -3 dB point. Typical bandwidth ranges for commercially available probes are, *e.g.,* 100 kHz $-$ 50 MHz or 3 MHz $-$ 3 GHz.

Note that, since the strength of EM fields rapidly decreases with the distance from the originating source, *i.e.,* the amplitude decreases in proportion to the inverse cube of the distance $\frac{1}{d^3}$ in the near-field, ICs can be decapsulated in order to position EM probes as close as possible to the die's surface and improve the SNR of EM measurements.

2.2.3 IC Decapsulation

The operation of opening the plastic package of an IC is called decapsulation. A relatively cheap way to decapsulate an IC is by chemical etching. This can be done either manually or with the aid of professional equipment, which is typically more costly, but also more precise.

Decapsulation usually consists of the following steps[1]:

[1]Etching should be performed in laboratory with the necessary equipment and according to the safety norms regulating the usage of chemicals.

1. Mill a small cavity on the top surface of the packaged chip using a drill bit, *e.g.,* use Dremel rotary tools to create a sort of sink container for acid in order to avoid spreading it all over the package during the etching process.

2. Place the chip on a hotplate heated up to around 100 °C. This temperature point might vary depending on the particular chemicals and package materials involved.

3. Dispense a few drops of acid on the top surface of the package in order to corrode the plastic covering the die, *e.g.,* using fuming nitric acid (HNO_3).

4. When the package has been exposed adequately to the corrosive action of acid and the die is visible, then it can be cleaned with some acetone and dried out at room temperature.

5. Finally, the die can be inspected with the aid of a microscope to verify if there are no package' residuals on the die and the bonding wires are intact. A picture of the die can be taken to identify those parts of the IC that are more relevant for side-channel analysis.

Note that, while it is possible to remove the bonding wires and re-wire the I/O to a convenient board for further analysis, this is typically a costly operation which requires a bonding machine. Hence, particular care has to be taken when dispensing the drops of acid and when cleaning the chip in order to avoid tearing off the bonding wires. Note that if the bonding wires are made of copper, then a mixture of nitric acid and Oleum should be preferably used in order to avoid corroding the wires with the acid.

Decapsulation is particular advantageous in case of EM measurements since it allows to:

1. Lower down the probe's coil until touching the passivation layer, thus decreasing the distance between the EM source and the probe, while ultimately increasing the SNR.

2. Isolate the activity of individual parts of the circuit which are more relevant for the analysis, *e.g.,* memories or cryptographic accelerators.

In order to take full advantage of decapsulated chips, high-resolution EM micro-probes can be employed.

2.2.4 EM Micro-probes

EM micro-probes allow to measure EM fields directly on the die surface at extremely high spatial resolutions, *e.g.*, with diameters between 100 µm and 150 µm. These probes allow to accurately pinpoint parts of the circuitry once the chip has been decapsulated and the die is visible. The relevant areas of the die, *e.g.*, memories and hardware accelerators, can be located and inspected with the aid of a microscope.

EM measurements involving EM micro-probes are typically performed using high-precision motorized positioning systems to automatically scan the die's surface. In order to isolate the EM field sources as much as possible, small measurement steps should be performed. However, note that reducing the measurement's step size increases the measurement's time for the same inspected area. Hence, a good trade-off between the size of the probe, its sensitivity, and the number of scanning steps must be typically found. One way to increase the SNR of EM measurements when employing high-resolution EM micro-probes is to use probes' pre-amplifiers.

2.3 Implementation Attacks

Implementation attacks are typically classified according to two main criteria that are orthogonal to each other [71].

The first criterion is related to the type of interaction that the adversary has with the cryptographic device under attack. This can be either *active* or *passive*.

- *Passive Attacks*:
  The adversary passively collects information about the secret key by observing some physical properties of the cryptographic device operating within its nominal levels of operation, *e.g.*, the adversary monitors the power consumption or EM field variations of an IC.
- *Active Attacks*:
  The adversary actively manipulates one or more parameters of a cryptographic device or its environment, like the power supply or temperature, in order to exceed the nominal levels of operation and obtain information about the secret key from its malfunctioning.

The second criterion classifies attacks depending on the degree of invasiveness that the adversary has in respect to the cryptographic devices under attack, *i.e.*, to which extent the adversary physically tampers with the target device.

These types of attacks can be either *non-invasive*, *semi-invasive*, or *invasive*.

- *Non-Invasive Attacks*:
  The adversary does not apply any manipulations (neither chemical or mechanical) to the cryptographic device, but only plays with the external logical interfaces of the chip. This type of attacks is typically very cheap to perform, and therefore they pose a serious threat to the security of cryptographic devices. Also, they typically leave behind no trace of the performed attack.

- *Semi-Invasive Attacks*:
  The adversary modifies the outer layers of the device, *e.g.*, by removing the chip's package, in order to gain access to the die's surface, but does not establish any electrical contacts with the chip, *i.e.*, the chip's passivation layer is neither removed or altered.

- *Invasive Attacks*:
  In this type of attacks, the adversary gains access to the metal layers or the die's substrate to either establish a direct contact with the circuit or to modify its functionality. Invasive attacks are the strongest type of attacks, but typically require industrial equipment, like probing stations, which are very expensive and hardly affordable.

The most common types of implementation attacks are Side-Channel Attacks (SCAs) and Fault Attacks (FAs). Prominent examples of SCAs are timing analysis attacks, power analysis attacks, and electro-magnetic analysis attacks. These types of attacks extract information about the secret key by observing either the time duration, power consumption, or electromagnetic radiation, respectively.

Timing attacks were introduced in 1996 by Paul C. Kocher [66], while power analysis attacks were subsequently introduced in 1999 by Paul C. Kocher, Joshua M. Jaffe, and Benjamin C. Jun [68][2]. Finally, electromagnetic attacks were introduced in 2001 by Quisquater and Samyde [96].

Side-channel attacks belong to the class of passive attacks and can be either non-invasive or semi-invasive, *e.g.*, when the chip's package is removed to access the chip's die with an EM probe.

Fault attacks try to recover the secret key by taking advantage of the erroneous behavior of a device during cryptographic computations. Fault attacks are active attacks which can be either non-invasive, semi-invasive, or invasive. The higher is the level of invasiveness, the higher are the costs to perform the attacks.

[2]Note that power analysis attacks were already described in a technical report by the same authors in 1998.

Table 2.2: Classification of implementation attacks according to the degree of invasiveness of the adversary, its type of interaction with the device, and the costs associated to mount successful attacks: side-channel attacks are highlighted in **blue**, while fault attacks are highlighted in **green**.

	Active	**Passive**	**Costs**
Non-invasive	Glitches Spikes	Timing attacks Power attacks EM Attacks (Far-field)	Low-Medium
Semi-invasive	Laser beam EM pulses	EM attacks (Near-field)	Medium
Invasive	Laser FIB	Probing	High

One method to inject faults is to insert spikes into the clock supply to corrupt data transfers between registers and memories. This type of attacks is called glitch fault injection attacks and they are non-invasive. By removing the package, semi-invasive attacks are possible, *e.g.*, using light sources or EM fields to produce faults [59, 62]. Using a laser cutter or a Focused Ion Beam (FIB) is possible to perform invasive attacks which permanently change the behavior of the chip.

Table 2.2 provides an overview on the classification of implementation attacks.

2.3.1 Side-Channel Attacks

Since the seminal work of Paul C. Kocher [66], side-channel attacks represent one of the major threats to the security of cryptographic devices due to their limited costs and high effectiveness. Side-channel attacks are typically based on a divide-and-conquer strategy in which small parts of a secret key, called sub-keys, are recovered iteratively.

Side-channel attacks can be classified according to different parameters. The most relevant ones are briefly surveyed here.

Source of Side-Channel Information Cryptographic implementations may leak information about secret computations over different side-channels. The most prominent sources of side-channel information are *execution time*, *power consumption*, and *EM fields*.

Timing attacks exploit key-dependent variations in the execution time of cryptographic algorithms, *e.g.*, during the computation of RSA-based digital signatures. This type of attacks can be avoided by executing all key-dependent computations in constant-time independently of the inputs. However, another source of side-channel time information might be due to the processors' cache mechanisms [120]. One possible way to get rid of such side-channels is to disable caches or to clear/fill them out before any cryptographic computations happen. However, this is not always possible for performance reasons.

Another source of side-channel information is associated to the power consumption of cryptographic devices. Power analysis attacks exploit key-dependent variations in the power consumption of a cryptographic device. In this case, making power consumption constant has been shown to be too costly and difficult to achieve in practice. Hence, other techniques operating at higher abstraction levels are typically preferred to counteract side-channel attacks based on power analysis.

Electric currents flowing into digital cells generate magnetic fields, hence EM fields represent yet another source of side-channel information that should be considered in practice. EM analysis attacks exploit key-dependent variations in the EM field (either far-field or near-field) of a cryptographic device.

Other side-channels exploited in literature are sound emissions, *e.g.*, generated by the vibration of capacitors and coils in voltage regulation circuits [52], and photon emissions, *e.g.*, photons emitted by CMOS devices [49, 107].

Side-channel information coming from multiple sources can be aggregated to increase the effectiveness and efficiency of attacks [3].

Simple and Differential Attacks Side-channel attacks are typically referred to as either *simple*, *e.g.*, Simple Power Analysis (SPA), or *differential*, Differential Power Analysis (DPA).

SPA attacks exploit the information from either a single or a few observations using the same input and try to *"derive the key more or less directly from a given trace"* [71, p. 101], *e.g.*, by visual inspection. SPA attacks are well known in the context of public-key cryptography, *e.g.*, the implementation of modular exponentiations using the Square-and-Multiply (S&M) algorithm can be attacked by SPA attacks: if the adversary is able to distinguish between a square and a multiplication step in a power trace, then the secret key can be recovered straightforwardly [68]. In practice, SPA attacks are typically limited by the fact that they require a high SNR in order to perform visual inspection and also the details of the implementation should be typically known to the adversary.

On the contrary, DPA attacks exploit the information from multiple traces using several inputs and statistical methods. In this case, the adversary uses the knowledge about the algorithm and the input (resp. output) to recover the secret key. The target of DPA attacks are key-dependent intermediate values of computation, which depend on a known input (resp. output) and a small part of the secret key. Based on different key guesses, the adversary can compute the hypothetical intermediate value for each key guess and observation.

Profiled and Unprofiled Attacks Side-channel attacks can be further classified as either *profiled* or *unprofiled*.

In profiled attacks, the adversary can characterize the leakage distribution of the target device using known inputs and keys. Profiled attacks take place in two phases:

1. *Profiling phase*:
   The leakage distribution of the device is estimated for a known key and saved into templates. This phase requires full control of a device similar to the one under attack.
2. *Matching phase*:
   The leakage of the device under test with an unknown key is compared the previously profiled distribution in order to retrieve the secret key material.

Template attacks [26, 3] as well as the stochastic approach by Schindler et al. [106] are well known examples of profiled attacks.

On the contrary, unprofiled attacks do not have a profiling phase and try to identify the correct key on-the-fly using a statistical distinguisher and a hypothetical model of the leakage information. DPA as originally proposed in [68] uses the difference of means as a statistical distinguisher. Correlation Power Analysis (CPA) [22] uses the Pearson's correlation coefficient to find out linear dependencies between the measurements and a hypothetical model of power consumption. Mutual Information Analysis (MIA) [53, 9] uses mutual information as a generic non-parametric distinguisher. Robust SCA [43] uses the least square method to approximate the leakage information.

For unprofiled attacks, the leakages are typically modeled using the Hamming weight of a secret intermediate value or the Hamming distance of two secret intermediate values. Other models include the single-bit model, the zero-value model, and the identity model [71, Sec. 3.3].

Assumptions about the Leakage Distribution The leakage is typically assumed to be drawn from a normal Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and standard deviation σ. Other more generic models, such as Gaussian mixture models [70], and unparametrized models, such histogram or kernel density estimation methods, have been used to estimate the leakage distribution [125]. In general, the more the model is generic, the less the attack is efficient and the more computational power is required to perform the attack.

Dimensionality of the Leakage Distribution Side-channel information can be either single-valued (*e.g.*, time measurements in timing attacks) or multi-valued (*e.g.*, power or EM measurements consist of many values recorded over time).

According to how many dimensions of the side-channel leakage are exploited, side-channel attacks are classified as either *univariate* or *multivariate*.

In general, multivariate attacks are more powerful because they exploit more information from the side-channel leakages, but also require more computational power than univariate attacks.

Data Complexity and Number of Measurements The data complexity defines the number of inputs that are under control of the adversary. In some cases, the data complexity is limited by design, *e.g.*, due to a cryptographic protocol or due to a deployed countermeasure. The data complexity has not to be confused with the number of measurements that an adversary can record.

The number of measurements corresponds to the number of acquisitions that an adversary is able to record, *e.g.*, with the aid of an oscilloscope. In general, multiple acquisitions of the same input can help averaging out the measurement's noise and improve the SNR.

Spatial Resolution The resolution of a side-channel attack can be either *global* or *local*: the adversary has access to the side-channel of a device in its entirety, *e.g.*, the adversary can only measure the global power consumption of an IC including all active peripherals, or is allowed to measure the activity of single components on the chip, *e.g.*, measure the activity of a memory component using near-field EM probes.

2.3.2 Fault Attacks

Fault attacks are well-known implementation attacks introduced in the mid-nineties by Boneh et al. to defeat the security of cryptographic systems [19]. Since their introduction, several fault attacks against both symmetric [14, 90, 83] and asymmetric [12, 19] cryptographic algorithms were proposed in literature.

The main idea behind fault attacks is to disrupt the normal operation of cryptographic devices in order to induce errors during cryptographic computations and exploit the faulty output to gain information about the secret key.

Faults can be injected into cryptographic devices using different injection techniques such as:

- Sudden transient changes in the power supply lines [7, 8];
- Glitches in the signal line, *e.g.*, clock or reset line [6];
- Strong EM radiations [108] as well as light sources such as ultraviolet lamps [108], camera flashes [59] and lasers [114];
- Bringing the device temperature outside the working ranges [63].

Fault attacks are active attacks which can be performed with different levels of invasiveness and spatial resolution (either global or local).

Non-invasive techniques include over- and under-voltage variations on the power supply (spike attacks), overheating or freezing the chip, and inserting glitches on the clock lines. The variation of these parameters, causes the transistors within the ICs to switch at a faster, resp. slower, rate than allowed, hence producing errors. These can be attributed to, *e.g.*, setup time violations.

Invasive techniques employ the use of expensive tools, that range from micro-probing stations to FIB machines. A more interesting class of attacks are semi-invasive attacks, like those using light sources or EM fields. This type of attacks typically produces more useful faults than those produced using non-invasive attacks, while keeping the equipment costs lower than those needed for invasive attacks.

A fault model describes the conditions for which fault attacks can successfully extract the key from faulty computations:

- *Spatial Location*:
  The spatial location specifies where the fault should be injected in order to be effective, *e.g.*, memory elements or data flow.

- *Temporal Location and Duration*:
  The temporal location specifies when the fault should be injected during the cryptographic computations, *e.g.,* in the penultimate round of AES, while the temporal duration specifies for how long the faults should be injected, *e.g.,* permanent or transient faults.

- *Fault Type and Width*:
  The fault type specifies how computations can be affected by the injections, *e.g.,* bit set/reset, bit flip, random fault, while the width specifies how many bits can be affected from the faults. In some cases, it might also be useful to specify whether any specific or arbitrary bits are affected by the faults.

- *Multiplicity*:
  The multiplicity specifies how many faults should be injected during one single cryptographic computation in order to mount a successful attack, *e.g.,* two or more fault injections can be needed if countermeasures against fault attacks are deployed.

The easier a certain fault model can be realized in practice, the more powerful is considered the fault attack. Fault attacks can be roughly classified into DFAs and Faulty Ciphertext-Only Fault Attacks (FCOFAs).

DFAs are chosen-plaintext attacks which have been introduced by Biham and Shamir in 1997 [14]. They exploit the differences between correct and faulty ciphertexts in order to retrieve information about the secret key by verifying invariant conditions during cryptographic operations. Note that, although the plaintexts can be chosen at random from the adversary, yet the adversary must be able to feed the same plaintext two times into the device in order to obtain a correct and a faulty ciphertext on the same input. One of the most powerful DFA against AES is that from Saha et al. [103].

FCOFAs were introduced in [51] and further investigated in [39]. This class of fault attacks is the most generic one, as it only assumes that a collection of (faulty) ciphertexts obtained under the same secret key is known to the adversary. In this case, the adversary injects a bias in the distribution of the intermediate values and extracts the secret key under the assumption that the bias can be identified from the faulty ciphertexts when the correct key is guessed.

2.4 Implementation Countermeasures

The implementation of cryptographic algorithms may leak information about the secret key over multiple side-channels and can also be subjected to fault attacks. In order to thwart side-channel and fault attacks, implementation countermeasures are typically deployed on cryptographic devices.

They can be classified according to two main criteria, which are orthogonal to each other.

The first criterion is the level of abstraction on which the countermeasure applies:

- *Physical Level*:
  This type of countermeasures acts at the physical level by adding either mechanical or electrical components on the cryptographic device, *e.g.*, fault injections can be detected by sensing changes in the gradient of light [6], while side-channel attacks can be made harder by using EM shields to limit the emissions of cryptographic chips [96].
- *Logic Level*:
  This type of countermeasures acts at the hardware level by changing the logic design style of CMOS gates, *e.g.*, custom logic styles such as Sense Amplifier Based Logic (SABL) [118] or standard logic styles such as Wave Dynamic Differential Logic (WDDL) [119] can be used to make the power consumption look independent to the adversary.
- *Algorithmic Level*:
  This type of countermeasures acts at the algorithmic level by modifying the implementation of cryptographic algorithms without changing the input/output characteristic, *e.g.*, random delays can be inserted to make attacks more difficult or the algorithm is randomized using secret sharing techniques in order to protect the secret keys.
- *Protocol Level*:
  This type of countermeasures wraps cryptographic primitives into protocols from which the security is derived, *e.g.*, frequent key updates can be used to prevent the adversary from obtaining information about the secret key.

The second criterion is the type of action that is performed by the countermeasure, *e.g.*, side-channel countermeasures aim at either hide, mask, or bound the leakage to a certain extent, while fault countermeasures aim at either detect or bound fault injections.

Side-channel countermeasures can be classified as follows:

- *Masking*:
  Masking schemes aim at making side-channel observations independent of the processed secret data at the adversary's eyes.
- *Hiding*:
  Hiding countermeasures aim at reducing the SNR of exploitable information along either the time or the amplitude dimension.
- *Re-Keying*:
  Re-keying schemes aim at performing frequent key updates in order to limit the information that can be leaked under a certain key.

Typically the costs associated to the design, testing, and deployment of countermeasures is higher at lower levels of abstractions. Also, at the lower levels the countermeasures are based on assumptions which are harder to meet in practice, *e.g.*, perfect symmetries and propagation delay times in protected logic styles can be hardly achieved due to physical processing variations.

2.4.1 Masking Schemes

Masking schemes were introduced independently in 1999 by Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi at CRYPTO [24] and by Louis Goubin and Jacques Patarin at CHES [57] as a countermeasure against side-channel attacks.

A masking scheme describes a way to share input data, a procedure to perform computations on shared data, and a way to reconstruct output data from shared output data. A masking scheme must ensure that all masks are uniformly distributed and independent of the secret, each masked intermediate value is uniformly distributed independently of the secret intermediate state, and that each secret is shared during the full duration of a cryptographic operation, *i.e.*, no partial reconstructions should occur before the shared output is computed.

A d^{th}-order masking scheme, which is correctly designed and implemented, allows to withstand up to d^{th}-order attacks, *i.e.*, :

- Higher-order statistical moments of the leakage distribution have to be estimated in order to successfully reveal the secret, *i.e.*, the $(d+1)^{\text{th}}$ statistical moment should be estimated at least;
- The dimension of the leakage distribution increases with the masking order,

i.e., $(d+1)$ Points of Interests (POIs) have to be selected in order to successfully retrieve the secret key.

The work of [24] showed that the number of measurements required to mount a successful DPA increases exponentially with the number of shares in presence of Gaussian noise $\mathcal{N}(\mu, \sigma^2)$, *i.e.*, the number of measurements N required to distinguish a bit value is lower bounded by $N \geq \sigma^{d+4\frac{\log p}{\log \sigma}}$, when the leakage of $(d+1)$ shares associated to a single bit with Bernoulli distribution $\mathcal{B}(p)$ is given.

The main difficulty when designing masking schemes is how to perform shared computations on non-linear functions efficiently, *e.g.*, S-boxes. In fact, while the shared computation of linear functions is typically straightforward, as the original linear function can be applied to each share individually, most of the issues arise with non-linear functions, as they typically require interactions among the shares. For this reason Chapter 3 and Chapter 4 will focus on protecting the S-box operation, which is the only non-linear operation in AES.

Boolean Masking Boolean masking uses the bitwise XOR operation $\oplus$ over the binary field GF(2) such that:

$$x = x_m \oplus m_{x,1} \oplus \ldots \oplus m_{x,s-1},$$

where $(m_{x,i})_{1 \leq i \leq s-1}$ are randomly generated masks and x_m is the masked value of the secret intermediate value x. A 1st-order Boolean masking scheme for AES is illustrated in [71, p. 230].

Arithmetic Masking Arithmetic masking uses either addition or multiplication in the modular ring $\mathbb{Z}_n$, as follows:

$$x = \left(x_m + \sum_{i=1}^{s-1} m_{x,i} \right) \bmod n \quad \text{or} \quad x = \left(x_m \cdot \prod_{i=1}^{s-1} m_{x,i} \right) \bmod n$$

Some cryptographic algorithms like SHA-2 use both arithmetic and Boolean operations, thus requiring to switch between arithmetic and Boolean masking schemes [31, 56].

The Issue of Glitches In practice, many implementations of masking schemes are vulnerable due to the glitches arising in digital circuits. Glitches are temporary transitions that often occur at the output of logic cells due to different propagation times in digital circuits, *i.e.*, the output of logic gates can change several times before they get to a final stable value. There are many reasons behind the insurgence of glitches, *e.g.*, the input wires of a cell might have different capacitances or they might traverse different paths, hence arriving at different points in time.

Glitches pose a serious threat to the security of masking schemes as exemplary shown in [72]. The first effort to counteract glitches is due to [50]. Today, glitches can be avoided by employing either glitch-free logic styles or glitch-resistant masking schemes.

There are basically four approaches to glitch-resistant masking schemes: Threshold Implementations (TIs), Polynomial Masking Schemes (PMSs), Consolidated Masking Schemes (CMSs), and Domain Oriented Maskings (DOMs).

Threshold Implementations TIs were introduced in 2006 by Svetla Nikova, Christian Rechberger, and Vincent Rijmen [84]. TIs are based on secret sharing techniques and are provably secure in presence of glitches.

Let f be a vector function $(z^1, \dots, z^q) = f(x^1, \dots, x^p)$ and $\mathbf{x}^j = (x_i^j)_{1 \le i \le s}$ denote the vector of shares associated to the input x^j. Then, a 1st-order TI of f is obtained by splitting the function f into s functions $(f_i)_{1 \le i \le s}$ such that the following three properties hold:

1. *Correctness*:
   Both input and output shares provide the same input and output values of the original function f once recombined:
   $$(z^1, \dots, z^q) = f(x^1, \dots, x^p) = \sum_{i=1}^{s} f_i(\mathbf{x}^1, \dots, \mathbf{x}^p),$$
   where $x^j = \sum_{i=1}^{s} x_i^j$ for all $1 \le j \le p$.
2. *Non-completeness*:
   Every f_i is independent of *at least* one share of each input variable. Let $\mathbf{x}_i^j$ denote the reduced vector of shares $(x_1^j, \dots, x_{i-1}^j, x_{i+1}^j, \dots, x_s^j)$. Then, without loss of generality, the property is satisfied by requiring that z_i is independent of x_i^j for all $j : 1 \le j \le p$:
   $$z_i = f_i(\mathbf{x}_i^1, \dots, \mathbf{x}_i^p), \quad \forall i : 1 \le i \le s.$$
   A realization of this property is easy to find in case f is a linear function, but it is typically more difficult in case f is a non-linear function. In fact, the minimum number of shares to implement a product of D variables satisfying correctness and non-completeness requires that $s \ge D + 1$. Hence, the minimum number of shares to realize a 1st-order TI of a non-linear function is 3 [84, Th. 3].

3. *Uniformity*:
   The output shares must have full entropy, provided that the input shares are uniform. In case the output shares are not uniform, then either fresh masks can be added to the output or a new sharing with a uniform output must be found in order to guarantee 1st-order security. This latter option requires to use an increased number of shares. Uniformity has the advantage that functions can be securely composed without introducing further randomness if the property is preserved through the operations.

There are several works which present 1st-order TIs, *e.g.*, area-optimized TIs of AES are illustrated in [82, 15], while [69] shows how to enable TIs for all 4-bit S-boxes with 3 shares. TIs of 5-bit and 6-bit S-boxes are further considered in [17].

Higher-Order Threshold Implementations (HOTIs) were introduced by Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzisalv Nikov, and Vincent Rijmen in 2014 [16]. The core idea is to extend the property of "non-completeness" to the d^{th}-order, *i.e.*, any combination of up to d functions f_i must be independent of at least one input share. However, it was shown in [97] that the composition of uniform sharings without refreshing is not necessarily higher-order secure. This means that fresh masks can be necessary to achieve higher-order security when composing HOTI functions.

Consolidated Masking Schemes The work of [97] introduced Consolidated Masking Schemes (CMSs) as an extension of TIs. CMSs inherit all properties of TIs and further guarantee provable higher-order security with $d+1$ shares, independently of the algebraic degree t. In fact, it was longly believed that higher-order security for a function having algebraic degree t could only be achieved using $s \geq td+1$ shares. A 2nd-order TI of AES in the form of CMS using $d+1$ input shares is provided in [35]. As a drawback, CMSs can not be easily scaled to any protection order.

Polynomial Masking Polynomial masking represents the first attempt to conceive both a glitch-resistant and higher-order masking scheme in literature. It is based upon Shamir's Secret Sharing Scheme (SSSS) and Multi-Party Computation (MPC) protocols [95]. Polynomial masking was introduced at CHES 2011, while a first prototype of AES on FPGA was presented at CHES 2013. Subsequently, a 1st-order and a 2nd-order polynomially masked implementation of the PRESENT S-box on FPGA were published in [36].

While TIs and CMSs are faster, smaller, and have considerably lower randomness requirements than PMSs, these latter still have interesting advantages over TIs, *e.g.*, they can be straightforwardly adapted to any block-cipher and inherently scaled to any masking order using regular hardware structures, *i.e.*, they can be implemented in full-custom secure logic and synthesized for different masking orders, hence making it possible to quickly adapt their side-channel resistance according to a required certification level. Additionally, they provide some basic resistance against fault attacks if less than $1/3$ of the shares are faulted [95].

PMSs will be the subject of Chapter 3 and Chapter 4.

Domain-Oriented Masking DOM is a new approach to masking introduced by Hannes Gross, Stefan Mangard, and Thomas Korak [58] which improves over TIs, by requiring less area and randomness as well as enabling inherent scaling to any protection orders.

The basic idea is to assign each share to a different "domain" and secure domain crossings, which is necessary in case of non-linear functions, by adding fresh randomness and registers to block the propagation of glitches.

There are mainly three differences between TIs and DOM: (1) DOM is a domain oriented approach, while TIs follow a function oriented approach, (2) the minimum number of shares $d+1$ is always used by DOM, hence leading to reduced area and randomness requirements when compared to TIs, (3) DOM protected designs can be scaled to any protection orders like in the case of PMSs.

2.4.2 Hiding Schemes

Hiding countermeasures aim at reducing the SNR of exploitable side-channel information. There are several techniques which can be deployed along either the time or amplitude dimension. While those techniques which consider the amplitude dimension are typically applied at the physical level, *e.g.*, by changing the power characteristic of the performed operations, those which consider the time dimension are typically applied at the algorithmic level, *e.g.*, by introducing time disarrangements during cryptographic operations.

Hiding countermeasures are typically more lightweight as compared to masking schemes, but also less effective in terms of security gain, *i.e.*, the security gain of hiding countermeasure is typically linear or quadratic. For this reason, hiding countermeasures are usually implemented in addition to masking schemes, and not alternatively to them [100]. Note that introducing timing disarrangements represents also a quite effective countermeasure against fault attacks, as they decrease the probability that a fault affects the same operation at a given point in time.

Two of the most prominent techniques along the time dimension are Random Delay Insertions (RDIs) and shuffling schemes. These latter have the advantage that they typically do not affect the performance figures of the original implementation.

Random Delay Insertion This technique involves inserting random delays during cryptographic computations such that measurements become misaligned and appear more noisy to the adversary, hence increasing the attack complexity. The insertion of random delays can be done either in hardware using Random Process Interrupts (RPIs) or in software by randomly inserting "dummy" operations during the execution of cryptographic computations, *e.g.*, `NOP` instructions [32]. The authors of [30] showed that the number of measurements, which are required for a successful attack against the RPI countermeasure, grows quadratically with the standard deviation of the introduced delays.

Shuffling Schemes Shuffling schemes aim at introducing noise in the measurements by randomly permuting the execution order of a set of independent secret operations such that the final output of the algorithm does not change, *e.g.*, the execution order of the S-box within AES can be permuted without affecting the final result. The permutation is typically obtained using either a Random Start Index (RSI) or a Random Permutation (RP). The RSI approach offers higher performance but also less security than the RP approach, hence this latter is typically preferred. For a more detailed treatment on shuffling schemes the interested reader is referred to [124].

2.4.3 Re-Keying Schemes

The success probability of most cryptanalytic attacks, like linear or differential cryptanalysis attacks, and implementation attacks, like side-channel and fault attacks, depends on the number of cryptographic operations q which can be observed by the adversary under the same secret key.

While for cryptanalytic attacks, q is typically a quite large value, in the case of implementation attacks this number is rather small, *e.g.*, a successful fault attack on AES can require only $q = 2$ encryptions under the same key (only one faulty pair) [103].

The seminal idea of using re-keying schemes as a countermeasure against side-channel and fault attacks dates back to the U.S. Patent from Paul Kocher in 2003 [67]: it consists of generating a new session key every $(q-1)$ cryptographic operations in order to bound the number of cryptographic operations which can be observed by the adversary under the same key. In this way, the adversary does not have enough information to mount a successful attack.

A re-keying scheme provides inherent security against multiple observation attacks (cryptanalytic and implementations attacks that require $q > 1$ observations) as long as the same key is generated only with negligible probability.

However, single-observation attacks like SPA-based template attacks (for which $q = 1$) can not be inherently prevented by re-keying schemes. Note that, although the success probability of single-observation attacks heavily depends on the amount of noise which is present in the measurements, the security of re-keying schemes might anyway be affected by single-observation attacks even in presence of noise, *e.g.*, using key ranking and enumeration algorithms [121].

According to [75], a re-keying function should satisfy the following properties:

Property 1. Each bit of the session key should depend on many bits of the master key.

Property 2. No additional key material should be used, other than the master key.

Property 3. Protecting the re-keying function against SPA and DPA attacks should be easy and more convenient than protecting the re-keyed cipher in terms of speed and size.

The work of [41] additionally introduced the following property:

Property 4. The re-keying function should be non-invertible such that forward security can be guaranteed.

Re-keying schemes can be classified in two categories depending on whether the re-keying function is stateful or stateless.

A stateful re-keying scheme generates a fresh key from previously stored secret information like a previous key or a secret index, while a stateless re-keying scheme generates a fresh key from a secret key and public nonce.

Stateful Re-keying A stateful re-keying scheme generates a fresh key from a previously stored secret key or secret index using a re-keying function. The new fresh-key is stored permanently, possibly overwriting the previous key.

Each party has to store the same initial key and an index counting how many keys have been derived already. Then, if the two parties want to communicate, they have to synchronize the keys by deriving new keys until the indexes of the two parties match and a common key is obtained.

There are mainly two approaches to match the indexes of the two parties: either one of the two parties synchronizes its key to match the other party's key or both parties try to "meet in the middle".

This latter approach typically requires an inverse re-keying function to roll back to previous keys. The state-of-the-art approach to key-derivation is based on a skip-list structure so that a session key can be efficiently derived in logarithmic time [67, 1].

The stateful approach does not suffer from the secure initialization problem [88], as fresh keys are permanently overwritten after $(q-1)$ operations. However, the stateful approach is typically very expensive, as it requires Non-Volatile Memory (NVM), and it is not practicable for many applications, as it requires long times for synchronizing the keys between the parties and re-programming the NVM after a new key has been derived.

Stateless Re-keying A stateless re-keying scheme or fresh re-keying scheme generates a fresh key from a secret master key and public random nonce using a fresh re-keying function.

Definition 1 (Fresh Re-keying Function) *A fresh re-keying function is a function* $g\colon \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$ *which takes a secret randomly chosen (but fixed) key* $k \in \{0,1\}^\kappa$ *and a public randomly chosen nonce* $r \xleftarrow{\$} \{0,1\}^n$ *to generate a private session key* $k^* \in \{0,1\}^\kappa$ *satisfying the previously stated properties.*

The stateless approach does not require an NVM, but it requires a secure initialization [88], *i.e.,* a mechanism to retain the key entropy when the device is repetitively reset to its initial state, and a Pseudo Random Number Generator (PRNG) to generate nonce values.

Note that the security of fresh re-keying schemes depends on the quality of the nonce values, *e.g.,* if the random source is biased, then the same session keys can be generated with higher probability, and on the mathematical properties of the re-keying function, *e.g.,* the same key should be generated only with low probability.

In this work, stateless re-keying will be considered in Chapter 6 and Chapter 7.

3 Design of a Higher-Order and Shuffled Polynomially Masked AES S-box

Contents

One of the most effective ways of protecting block-cipher implementations against side-channel analysis is to employ higher-order masking schemes.

The basic idea of higher-order masking schemes is to split the computation of secret-dependent intermediate values into multiple random shares, so to increase the adversary's data complexity exponentially with the number of shares [25, 94].

In the last years, a number of higher-order masking schemes have been proposed in literature along with statements of provable security. Nevertheless, they mostly turned out to be flawed or susceptible to side-channel attacks, *e.g.*, due to oversimplified modeling assumptions like in the case of glitches [72].

Polynomial masking [95, 101] is a d^{th}-order masking scheme based upon Shamir's Secret Sharing Scheme (SSSS) [111] and secure Multi-Party Computation (MPC) protocols [11].

Polynomial masking is particularly suitable for protecting implementations of cryptographic algorithms which are defined over finite fields, *e.g.*, the binary extension field $\mathsf{GF}(2^n)$.

In particular, any n-bit bijective S-box can be represented as a polynomial $\sum_{i=0}^{2^n-1} a_i x^i$ over $\mathsf{GF}(2^n)$ using Lagrange's interpolation formula. This representation allows to define any such S-box in terms of $\mathsf{GF}(2^n)$ operations only, for which polynomial masking can be applied straightforwardly.

In case of AES, the S-box can be computed as an inversion, followed by successive squarings, multiplications, and additions in $\mathsf{GF}(2^8)$. The inversion can be computed efficiently using *addition chains*, *i.e.*, by applying successive multiplications and squarings in $\mathsf{GF}(2^8)$.

This chapter introduces principles for the selection of addition chains, which lead to faster and more compact hardware implementations of S-boxes based on power functions in $\mathsf{GF}(2^n)$, *e.g.*, the AES S-box.

Based on these findings, we present the hardware design of a more efficient higher-order polynomially masked AES S-box which also includes a new shuffling countermeasure. Our shuffling countermeasure operates directly on the shares of the scheme during non-linear operations. It is lightweight and its effectiveness to hide higher-order univariate leakages will be shown in Chapter 5.

Parts of this chapter have been published at the "Theory of Implementation Security Workshop (TIs)" in 2016 [37].

Outline

- **Section 3.1** provides a formal description of polynomial masking.
- **Section 3.2** introduces the AES block-cipher and details on the AES S-box.
- **Section 3.3** introduces addition chains with a particular focus on star chains.
- **Section 3.4** provides new principles for the selection of implementation-efficient addition chains.
- **Section 3.5** describes the hardware design of a higher-order polynomially masked AES S-box based on our improved addition chains and presents our

proposed shuffling countermeasure.

- **Section 3.6** provides a summary of the chapter with a recapitulation of the main findings.

3.1 Polynomial Masking

Let $\mathsf{GF}(2^n)$ denote a binary extension field. A d^{th}-order masking scheme splits each secret intermediate value $x \in \mathsf{GF}(2^n)$ into a set of $m = 2d + 1$ random shares, such that no subset of cardinality $d < m$ reveals any information about the secret x.

A d^{th}-order masking scheme can be defeated by $(d+1)^{\text{th}}$-order side-channel attacks which exploit either the combined leakage of $(d+1)$ shares or the $(d+1)^{\text{th}}$ statistical moment of the leakage distribution.

The main operations of polynomial masking for any order $d > 0$ and secret intermediate value $x \in \mathsf{GF}(2^n)$ are summarized.

Initialization The scheme is initialized as follows: m field elements $(\alpha_i)_{1\leq i\leq m} \overset{\$}{\leftarrow} \mathsf{GF}(2^n)^*$ are chosen at random, such that $\forall i \neq j \in [1,m]$, $\alpha_i \neq \alpha_j$. Then, the m field elements $(\lambda_i)_{1\leq i\leq m}$ on the first row of the inverse of Vandermonde $(m \times m)$-matrix $(\alpha_i^j)_{1\leq i,j\leq m}$ are computed, as follows:

$$\lambda_i = \prod_{k=1,k\neq i}^{m} -\alpha_k(\alpha_i - \alpha_k)^{-1}.$$

The initialization elements $(\alpha_i, \lambda_i)_{1\leq i\leq m}$ are made public.

Sharing Let $P_x(Y) = x + \sum_{j=1}^{d} r_j Y^j$ be a random polynomial instantiated using d random coefficients $(r_j \overset{\$}{\leftarrow} \mathsf{GF}(2^n))_{1\leq j\leq d}$ and representing the secret intermediate value $x \in \mathsf{GF}(2^n)$. Then, the m random shares $(x_i)_{1\leq i\leq m}$ are obtained by evaluating P_x at the public points $(\alpha_i)_{1\leq i\leq m}$, as follows:

$$x_i = x + \sum_{j=1}^{d} r_j \alpha_i^j, \quad i \in [1,m].$$

Shared Addition The addition of two secrets $z = x + y$ can be performed independently on the shares $(x_i, y_i)_{1\leq i\leq m}$ in a straightforward manner, as follows:

$$z_i = (x_i + y_i) + \sum_{j=1}^{d} (r_j + r'_j)\alpha_i^j, \quad i \in [1,m], \tag{3.1}$$

where $P_x(Y) = x + \sum_{j=1}^{d} r_j Y^j$, $P_y(Z) = y + \sum_{j=1}^{d} r'_j Z^j$, and no fresh randomness is required.

Shared Multiplication by a Constant Similarly to addition, the multiplication of a secret intermediate value x by a constant value $c \neq 0$ can be performed independently on each share, as follows:

$$z_i = (cx_i) + \sum_{j=1}^{d} (cr_j)\alpha_i^j, \quad i \in [1, m], \tag{3.2}$$

where $P_x(Y) = x + \sum_{j=1}^{d} r_j Y^j$ and no fresh randomness is required.

Shared Multiplication The multiplication of two secrets $z = xy$ can be performed in three steps using the method described by Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson in 1988 [11]:

$$\begin{cases} \texttt{Step 1.)} & t_i & = & x_i y_i \\ \texttt{Step 2.)} & q_{i,k} & = & t_i + \sum_{j=1}^{d} s_j \alpha_k^j \quad , \quad i, k \in [1, m]. \\ \texttt{Step 3.)} & z_i & = & \sum_{w=1}^{m} q_{w,i} \lambda_w \end{cases} \tag{3.3}$$

First, the shares are pairwise multiplied in a straightforward way (`Step 1`). Then, the result is re-shared using d freshly generated masks $(s_j)_{1 \leq j \leq d}$ for each share (`Step 2`). Finally, the resulting polynomial is reduced to the original degree d by Lagrange interpolation using the public values $(\lambda_i)_{1 \leq i \leq m}$ (`Step 3`).

Reconstruction A secret intermediate value x can be reconstructed back from its shares using Lagrange's interpolation formula and the previously computed public values $(\lambda_i)_{1 \leq i \leq m}$, as follows:

$$x = \sum_{i=1}^{m} x_i \lambda_i.$$

3.2 AES

Rijndael is a block-cipher developed by Joan Daemen and Vincent Rijmen, and submitted to the public selection process of the Advanced Encryption Standard (AES) announced by the U.S. National Institute of Standards and Technology (NIST) in 1997.

In 2001, the U.S. NIST publicly announced the Federal Information Processing Standard (FIPS) 197 document [85], which specifies AES as a replacement for the Data Encryption Standard (DES).

The FIPS 197 specifies three instances of the Rijndael block-cipher [33] named AES-128, AES-192, and AES-256. The block-size of all three instances is fixed at 128-bit, while the key size is either 128-bit, 192-bit, or 256-bit, respectively.

Since 2002, AES is officially a standard of the U.S. federal government. According to [54], *"Rijndael was selected because it had the best combination of security, performance, efficiency, and flexibility"*.

While AES was originally intended for use by non-military U.S. government agencies to protect sensitive but not classified information, AES is currently included in international standards like ISO/IEC 18033-3 standard and approved by the National Security Agency (NSA) to protect top secret information.

The AES block-cipher is a Substitution-Permutation Network (SPN), which iterates a round function over a 128-bit state for either 10, 12, or 14 times, depending on the key size (128-bit, 192-bit, or 256-bit, respectively).

The round function consists of three layers that perform one or more transformations of the input: the byte substitution layer performs the SubBytes transformation, the diffusion layer performs the ShiftRows and MixColumns transformations, and the key addition layer performs the AddRoundKey transformation.

The SubBytes operation mainly introduces confusion by a non-linear transformation of the state and is also commonly referred to as the "AES S-box". It is the only non-linear operation in the block-cipher and also the most resource consuming operation to be masked in AES. For this reason, this chapter is focused on the AES S-box. In the remainder of this thesis, the term "AES algorithm" is used to specifically refer to the AES-128 encryption algorithm, if not otherwise specified.

3.2.1 The **SubBytes** Operation

The AES S-box is defined as the composition of two operations: a multiplicative inversion in the binary extension field $\mathsf{GF}(2^8) \simeq \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$ followed by an affine transformation in the binary field $\mathsf{GF}(2)$.

Using Lagrange interpolation, the AES S-box $\mathsf{S} : \{0,1\}^8 \to \{0,1\}^8$ can be conveniently expressed as the composition of $\mathsf{GF}(2^8)$ operations only (cf. [34, Sec. 10.4]):

$$\gamma = \mathsf{S}(\nu) = \delta_0 + \sum_{k=1}^{8} \delta_k \nu^{255-2^k} = \delta_0 + \sum_{k=1}^{8} \delta_k (\nu^{-1})^{2^k}, \tag{3.4}$$

where $\nu^{-1} = \nu^{254}$ and the coefficients $\delta = (\delta_k)_{0 \le k \le 8}$ are specified by the following vector:

$$\delta = (\texttt{0x63,0x05,0x09,0xF9,0x25,0xF4,0x01,0xB5,0x8F}).$$

It follows that the computation of $\mathsf{S}(\nu)$ can be performed as an exponentiation ν^{254} in $\mathsf{GF}(2^8)$, followed by successive squarings, multiplications (by constants), and additions in $\mathsf{GF}(2^8)$. This representation of the AES S-box allows for a straightforward application of polynomial masking as described in Section 3.1.

3.3 Addition Chains

The complexity of exponentiation in finite fields represents an important problem in cryptography. Addition chains provide an efficient way to compute exponentiations in finite fields. In particular, the inversion $\nu^{-1} = \nu^{254}$ in $\mathsf{GF}(2^8)$ can be computed efficiently using addition chains.

Definition 2 (Addition Chain) *An addition chain $\mathcal{C}$ for a positive integer q is a sequence of positive integers $\mathcal{C} = (a_0 = 1, \ldots, a_\ell = q)$, such that for every $1 \le i \le \ell$, there exist $0 \le j, k < i$ and $a_i = a_j + a_k$, where ℓ is the length of the chain.*

The length of the chain ℓ corresponds to the number of multiplication steps required to compute x^q. Note that every addition $a_i = a_j + a_k$ actually corresponds to a multiplication and every doubling $a_i = 2a_j$ actually corresponds to a squaring operation in the field.

3.3.1 Star Chains

Addition chains in the form $a_i = a_{i-1} + a_k$ are called *star chains* [21]. This type of addition chains is particularly suitable for compact hardware implementations, as the operand a_{i-1} is directly available from the previous multiplication step, hence it does not require any extra storage.

In general, finding the shortest addition chain for q in an efficient way is not an easy task. Over the years, different algorithms have been proposed to find short addition chains for a given integer q.

Algorithm 1 provides a naïve exhaustive search algorithm to find all addition chains for q in the form $a_i = a_{i-1} + a_k$ up to a given length ℓ. The algorithm runs in $\mathcal{O}(\ell!)$ steps and proceeds recursively by constructing a tree of all possible star chains of length ℓ.

Algorithm 1 Exhaustive Search Algorithm for Star Chains.

Require: $r, q \in \mathbb{N}^+, \ell \in \mathbb{N}, \mathcal{C} = (a_0 = 1, \ldots, a_{r-1})$.
Ensure: $\mathcal{S} = \{(a_0 = 1, \ldots, a_{r-1} = q) : \forall\, 0 < r \leq \ell\}$.

```
1:  procedure FindChain($q, \ell, \mathcal{C}$)
2:      $r = |\mathcal{C}|$                                        ▷ Find length of $\mathcal{C}$
3:      if $a_{r-1} = q$ then
4:          Save($\mathcal{C}$)                    ▷ Store $\mathcal{C} = (a_0 = 1, \ldots, a_{r-1 \leq \ell} = q)$ in $\mathcal{S}$
5:      else if $r \leq \ell$ then
6:          for $k = 0$ to $r - 1$ do
7:              $\mathcal{C} \leftarrow (a_0, \ldots, a_{r-1}, a_r = a_{r-1} + a_k)$     ▷ Add a new element $a_r$ to $\mathcal{C}$
8:              FindChain($q, \ell, \mathcal{C}$)
9:          end for
10:     end if
11: end procedure
```

3.4 New Selection Principles

Given a positive integer q, the addition chain with the smallest number of additions is selected among the set of all possible addition chains for q, in order to enable fast computations.

One additional selection principle, which is typically used in the context of cryptographic implementations, is to select those addition chains which maximize the number of doubling operations. In fact, squaring operations perform typically faster than multiplications on the same input values. However, increased area costs must be accounted for a dedicated squaring routine. Yet, there are typically many addition chains that satisfy these conditions for a given q.

Hence, we introduce two additional principles to improve the selection of addition chains geared towards performance:

Principle 1 *Select those addition chains which generate intermediate values that can be useful for subsequent operations within the considered cryptographic algorithm.*

Principle 2 *Select those addition chains which minimize the number of intermediate values that must be stored for exponentiation.*

Clearly, those addition chains fulfilling Principle 1 and Principle 2 provide two additional benefits, which are advantageous for efficient implementations: they allow for an overall speed-up of cryptographic implementations, by computing values which are needed in successive operations (Principle 1), and for a minimization of the storage required to save intermediate values during exponentiations (Principle 2).

Note that, while Principle 1 is algorithm specific (its applicability depends on the specific considered algorithm), Principle 2 is generally valid for any addition chains and of independent interest.

3.4.1 Previous Work

In recent works [99, 65, 80], the inversion $x^{-1} = x^{254} \in \mathsf{GF}(2^8)$ was implemented using the addition chain $\mathcal{C}^{254}_{\mathsf{orig}} = (1, 2, 3, 6, 12, 15, 30, 60, 120, 240, 252, 254)$.

This addition chain is graphically illustrated in Figure 3.1: each circle contains the results of the addition of two previous values. Since these values represent exponents, adding two values is equivalent to multiplying two powers with the same base.

It requires a total of 11 operations, of which 7 are squaring operations and 4 are multiplications, and to store the intermediate values x, x^2, x^3, and x^{12} in order to compute $x^{-1} = x^{254}$.

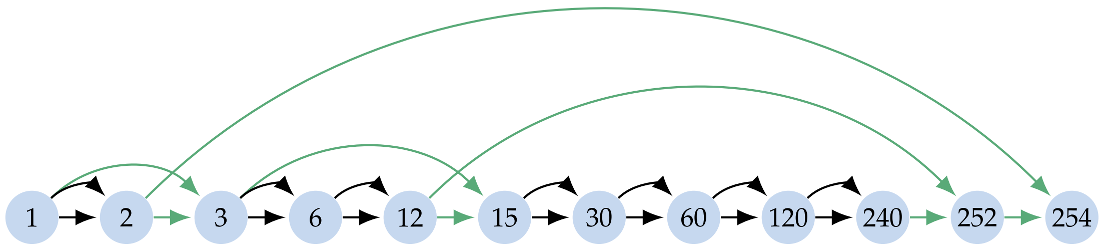


Figure 3.1: Inversion in $\mathsf{GF}(2^8)$ using $\mathcal{C}^{254}_{\mathsf{orig}}$. It requires 7 multiplication and 4 squaring steps, namely: $\mathbf{S} \to \mathbf{M} \to \mathbf{S} \to \mathbf{S} \to \mathbf{M} \to \mathbf{S} \to \mathbf{S} \to \mathbf{S} \to \mathbf{S} \to \mathbf{M} \to \mathbf{M}$. The intermediate values x, x^2, x^3, and x^{12} must be stored to compute $x^{-1} = x^{254}$. Multiplication steps are highlighted in **green**.

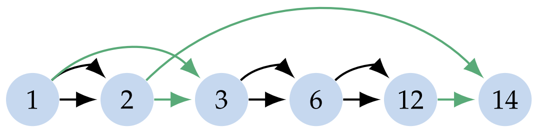


Figure 3.2: Inversion in $\mathsf{GF}(2^4)$ using $\mathcal{C}^{14}_{\mathsf{orig}}$. It requires 2 multiplication and 3 squaring steps, namely: $\mathbf{S} \to \mathbf{M} \to \mathbf{S} \to \mathbf{S} \to \mathbf{M}$. The intermediate values x and x^2 must be stored to compute $x^{-1} = x^{14}$. Multiplication steps are highlighted in **green**.

Similarly, the addition chain $\mathcal{C}^{14}_{\mathsf{orig}} = (1, 2, 3, 6, 12, 14)$ was used in [65] to perform the inversion in $\mathsf{GF}(2^4)$, as required by tower field implementations of AES. The addition chain $\mathcal{C}^{14}_{\mathsf{orig}}$ is graphically illustrated in Figure 3.2. It requires 5 operations consisting of 3 squarings and 2 multiplications and to store the intermediate values x and x^2 to compute $x^{-1} = x^{14}$.

Note that the initial value x must always be stored at least once in order to be able to compute x^2, therefore it is neglected in the remainder of this chapter.

3.4.2 New Addition Chains

The addition chain $\mathcal{C}^{254}_{\mathsf{orig}}$ requires the storage of 3 intermediate values, namely x^2, x^3, and x^{12}, and does not generate any intermediate value that can be useful for subsequent computations of AES, *e.g.*, it does not generate any of the values x^{127}, x^{191}, x^{223}, x^{239}, x^{247}, x^{251}, or x^{253} which are used for the affine transformation of the AES S-box, cf. Equation (3.4).

By running Algorithm 1, we found $|\mathcal{S}| = 6966$ addition chains for $(q = 254, \ell = 11)$, of which 754 were using 7 squaring and 4 multiplications, of which 106 required storage of only one single intermediate value (other than x), of which 55 also generated the value 127, that can be used during the AES affine transformation. Out of this last subset, we choose the addition chain $\mathcal{C}^{254}_{\mathsf{new}} = (1, 2, 4, 8, 9, 18, 36, 54, 108, 126, 127, 254)$ arbitrarily without considering any further constraints. The addition chain $\mathcal{C}^{254}_{\mathsf{new}}$ is shown in Figure 3.3 and has the following properties:

1. it consists of 11 multiplication steps, of which 7 are squaring operations and 4 are multiplications;
2. it produces the value x^{127}, which can be used by the AES affine transformation;
3. additionally to x, only one intermediate value has to be stored, namely x^{18}.

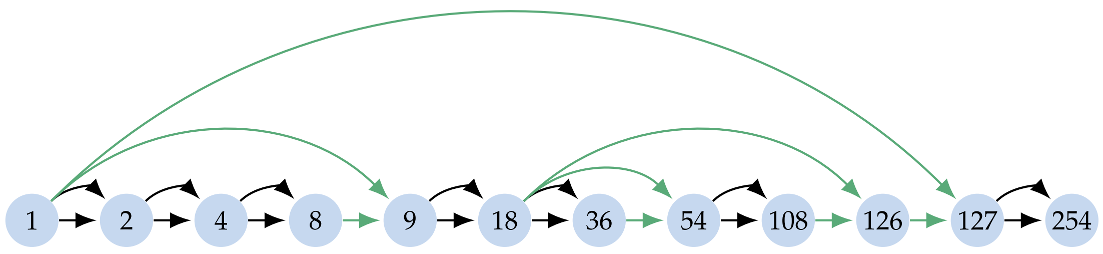


Figure 3.3: Inversion in $\mathsf{GF}(2^8)$ using $\mathcal{C}^{254}_{\mathsf{new}}$. It requires 7 multiplication and 4 squaring steps, namely: $\mathbf{S} \to \mathbf{S} \to \mathbf{S} \to \mathbf{M} \to \mathbf{S} \to \mathbf{S} \to \mathbf{M} \to \mathbf{S} \to \mathbf{M} \to \mathbf{M} \to \mathbf{S}$. Only the intermediate values x and x^{18} must be stored to compute $x^{-1} = x^{254}$. Multiplication steps are highlighted in **green**.

Hence, using the addition $\mathcal{C}^{254}_{\mathsf{new}}$ in place of $\mathcal{C}^{254}_{\mathsf{orig}}$ leads to faster and more compact AES S-box implementations:

1. the affine transformation is $1/7$ faster, as one squaring operation can be skipped;
2. the inversion is $1/3$ smaller, as only one intermediate value has to be stored instead of three values.

Note that, despite this seemingly small improvement, the area saving of hardware implementations might be significant when addition chains are masked, *i.e.*, when the intermediate values are shared and stored multiple times.

Similarly, we searched for star chains for $(q = 14,\ \ell = 5)$, which can be useful for efficient implementations of AES in tower fields, where the inversion $x^{-1} = x^{14} \in \mathsf{GF}(2^4)$ is used. By running Algorithm 1, we found 14 star chains for $(q = 14, \ell = 5)$, of which 8 were using 3 squaring and 2 multiplications, of which only 1 addition chain did not require the storage of any intermediate value, namely $\mathcal{C}^{14}_{\mathsf{new}} = (1, 2, 3, 6, 7, 14)$, as illustrated in Figure 3.4.

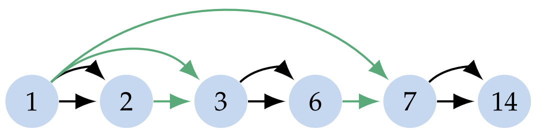


Figure 3.4: Inversion in $\mathsf{GF}(2^4)$ using $\mathcal{C}^{14}_{\mathsf{new}}$. It requires 2 multiplication and 3 squaring steps, namely: $\mathbf{S} \to \mathbf{M} \to \mathbf{S} \to \mathbf{M} \to \mathbf{S}$. No additional intermediate value must be stored. Multiplication steps are highlighted in **green**.

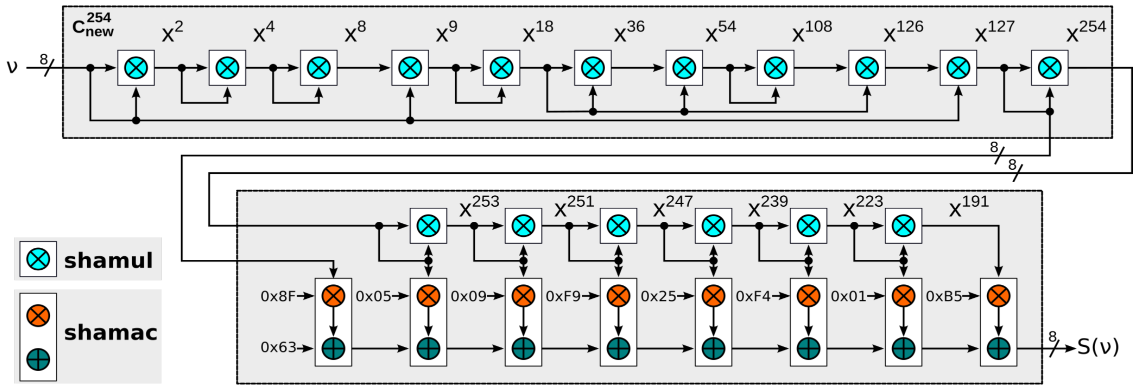


Figure 3.5: Evaluation of a polynomially masked AES S-box according to Equation (3.4) and using the addition chain $\mathcal{C}_{\text{new}}^{254}$ for inversion: The shared multiplication is highlighted in **light blue**, the shared multiplication by a constant is highlighted in **orange**, the shared addition is highlighted in **dark blue**. The upper gray box includes the steps for performing the inversion $x^{-1} = x^{254}$, while the lower gray box performs the affine transformation in $\mathsf{GF}(2^8)$.

3.5 Hardware Design

This section details the hardware design of a polynomially masked AES S-box for any order $d = (m-1)/2$. The design has an 8-bit interface, implements a single S-box instance, and consists essentially of three modules:

1. the "shared multiplication" module (`shamul`), which performs the multiplication of two shared secrets according to Equation (3.3);
2. the "shared multiplication by a constant and addition" module (`shamac`), which performs the multiplication of a secret by a constant value followed by a secret addition according to Equation (3.2) and Equation (3.1), respectively;
3. the "shuffling" module (`shuffle`), which generates random permutations from the symmetric group S_m to shuffle the activation order of the m secret shares within every other shared module.

The processing of a shared AES S-box using the addition chain $\mathcal{C}_{\text{new}}^{254}$ is illustrated in Figure 3.5.

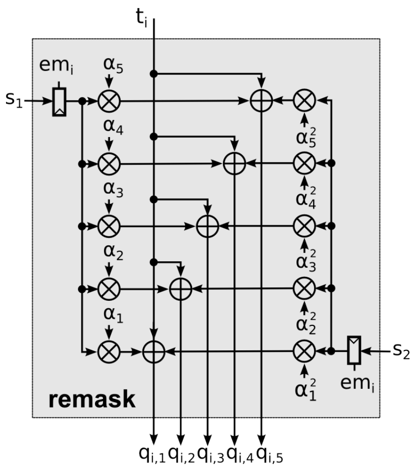


Figure 3.6: 2nd-Order `remask` Module for the i^{th} Share according to `Step 2.` of Equation (3.3). This module must be instantiated $m = 5$ times for $d = 2$.

The `shamul` Module The shared multiplication module (`shamul`) is used to perform the inversion defined by $\mathcal{C}_{\text{new}}^{254}$ and to iterate the squaring operations as needed by the affine transformation in $\text{GF}(2^8)$, cf. Equation (3.4) and Figure 3.8 for the case $d = 2$.

It basically implements the multiplication of two shared secrets according to Equation (3.3) using three submodules: the re-masking module (`remask`), as illustrated in Figure 3.6 for the case $(d, m) = (2, 5)$, the reconstruction module (`recon`) and a $\text{GF}(2^8)$ multiplier, as illustrated in Figure 3.7 for the case $(d, m) = (2, 5)$. These modules are implemented m times within the `shamul` module, one for each share $1 \leq i \leq m$.

The first step of the shared multiplication is performed by the $m \times \text{GF}(2^8)$ multipliers instantiated in the `shamul` module. The second step is implemented by the m `remask` modules, each one instantiating $m \times d$ $\text{GF}(2^8)$ multipliers and $m \times d$ $\text{GF}(2^8)$ adders. This step requires $d \times$ 8-bit of freshly generated random masks $(s_j)_{1 \leq j \leq d}$ for each share. Finally, the third step is implemented by the $m\times$ `recon` modules, each one instantiating $m \times \text{GF}(2^8)$ multipliers and $(m - 1) \times \text{GF}(2^8)$ adders.

Note that each one of the `recon` modules is additionally equipped with $m \times$ 8-bit D-type Flip-Flop (DFF) registers to block the glitches, arising from the different `remask` modules, joining together in the combinational paths, as well as $2 \times$ 8-bit DFF registers for storing the intermediate results of the inversion.

Due to the usage of our selected addition chain $\mathcal{C}_{\text{new}}^{254}$, the total number of DFFs in the circuit is reduced from $m \times 40$ to $m \times 16$, when compared to [80].

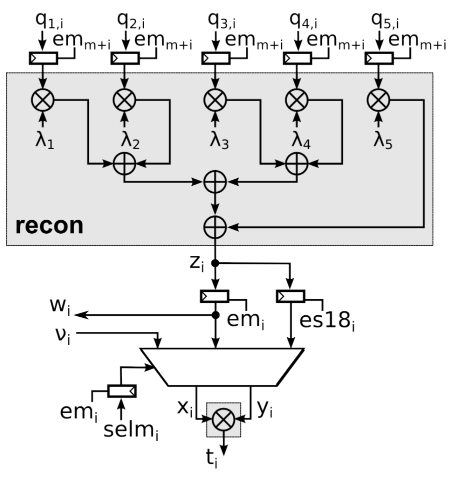


Figure 3.7: Upper part: 2nd-Order `recon` Module for the i^{th} Share according to `Step 3.` of Equation (3.3). This module must be instantiated $m = 5$ times for $d = 2$. Lower part: $\mathsf{GF}(2^8)$ multiplier according to `Step 1.` of Equation (3.3).

For the same reason, also the size of the multiplexers (MUXes) selecting the input shares (x_i, y_i) to the $\mathsf{GF}(2^8)$ multipliers is significantly reduced in practice. Finally, note that the switching activity of all MUXes in the design is clocked using "DFFs with Enable" as to avoid glitches on the select lines [80].

The `shamul` takes a total of $2 \times m$ clock cycles to perform the multiplication of two secrets and works as follows: in the first m clock cycles, the signals $\mathtt{selm}_\mathtt{i}$ for $i \in [1, m]$ select the proper inputs (where $\nu_\mathtt{i}$ represents the i^{th} input share of the S-box) to the $\mathsf{GF}(2^8)$ multipliers performing $t_i = x_i y_i$ and the enable signals $\mathtt{em}_\mathtt{i}$ for $i \in [1, m]$ are asserted to perform the first two steps of the shared multiplication.

The results are stored in the registers $\mathtt{q}_{\mathtt{j,i}}$ for $i, j \in [1, m]$. Then, during the next m clock cycles, the enable signals $\mathtt{em}_{\mathtt{m+i}}$ are asserted to finalize the shared multiplication.

Note that the enable signals $\mathtt{es18}_\mathtt{i}$ are activated only in the step in which the shared x^{18} value is computed. The output of the `shamul` are the shares $\mathtt{w}_\mathtt{i}$ for $i \in [1, m]$ which are sent to the `shamac` module.

The `shamac` Module The `shamac` module performs the shared multiplication by a constant followed by a shared addition according to Equation (3.2) and Equation (3.1), respectively. These operations do not require fresh randomness and can be performed independently on the individual shares, thus requiring only m clock cycles to be performed.

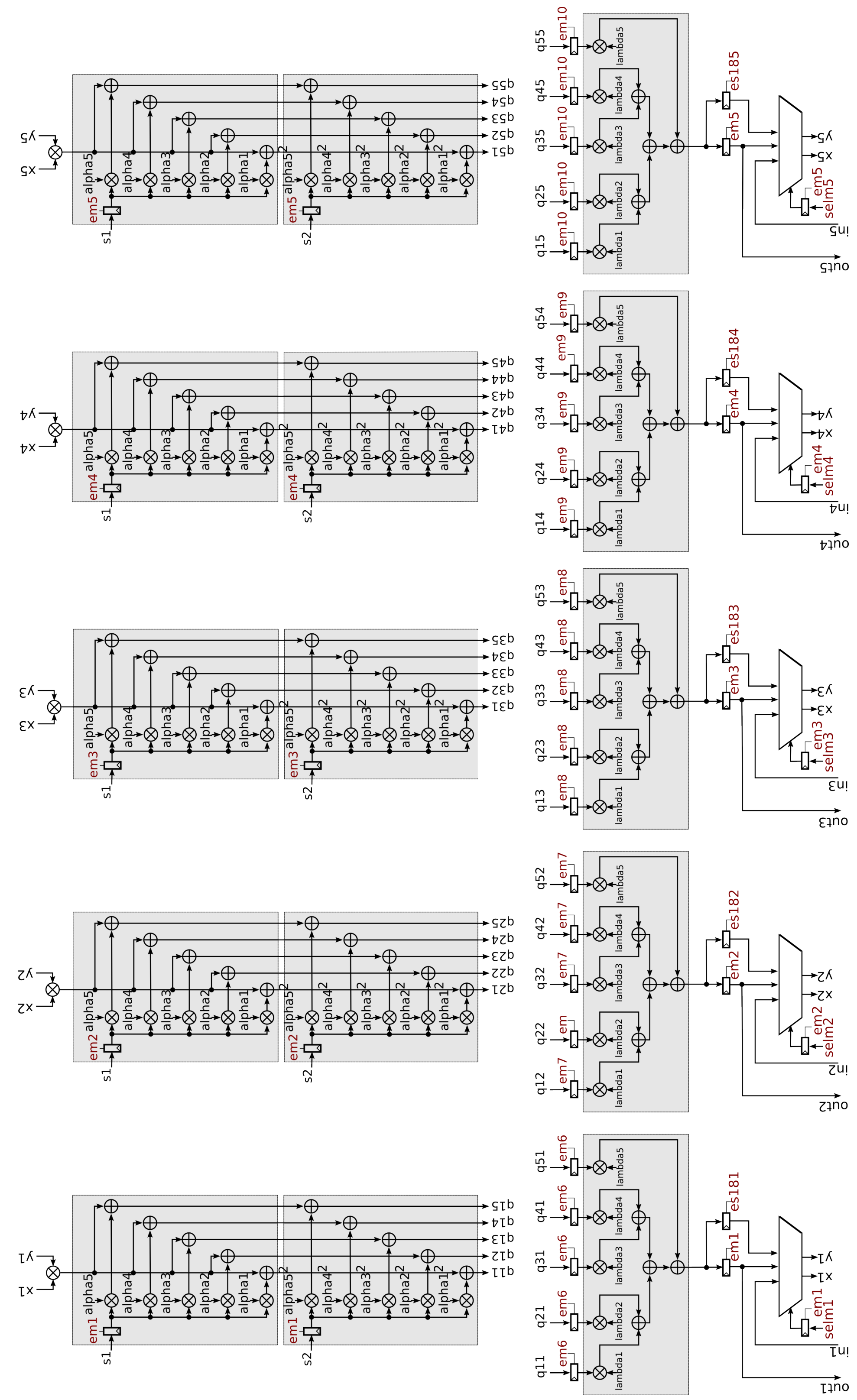


Figure 3.8: Shared GF(2^8) Multiplication Module for $(d, m) = (2, 5)$.

The `shamac` is used during the affine transformation, but it can be extended straightforwardly to support, *e.g.*, the initial sharing of the plaintext and key bytes, the initial `AddRoundkey`, and the `MixColumns+AddRoundkey` operations. One share of the `shamac` module is illustrated in Figure 3.9. Each share deploys $1 \times \mathsf{GF}(2^8)$ multiplier, $1 \times \mathsf{GF}(2^8)$ adder and $2 \times$ 8-bit DFF registers. The affine transformation uses the `shamac` module as follows: at the beginning the $\texttt{sel}_\texttt{i}$ selects the constant `0x63` from the $\texttt{MUX2}_\texttt{i}$, while the signal `selconst` selects the constant `0x8F` from the `MUX1` (common to all shares). Hence, the enable signal $\texttt{ea}_\texttt{i}$ is asserted and the result is stored in the register right below the adder. Finally, the $\texttt{MUX2}_i$ chooses the loop back input to continue the affine transformation using the constants (`0x05`, `0x09`, `0xF9`, `0x25`, `0xF4`, `0x01`, `0xB5`) selected by the signal `selconst` on the `MUX1`. Note that, in contrast to [80], we allow both the `shamul` and the `shamac` to be active at the same time during the affine transformation. This allows to save $7 \times m$ clock cycles per S-box computation, while not having any apparent impacts on the side-channel security of the implementation (cf. Chapter 5).

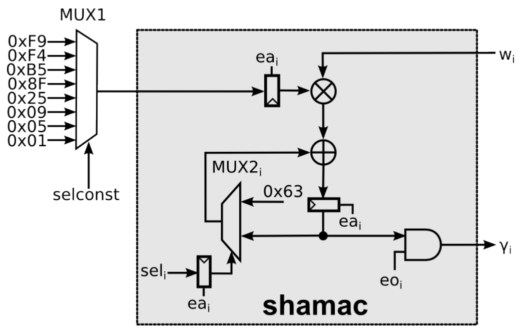


Figure 3.9: Multiplication and Shared Addition Module in $\mathsf{GF}(2^8)$ for the i^{th} Share: This module must be instantiated m times.

3.5.1 Adding Shuffling

Polynomial masking works by activating the secret shares individually, one after the other, such that only one secret share is active per clock cycle. For instance, the i^{th} share of the `shamac` module is activated when the m-bit enable signal $\texttt{ea}_\texttt{i}$ is set to logic '1'. This value is generated, *e.g.*, by a $\lceil \log_2 m \rceil$-bit counter, which is incremented every clock cycle and set to zero every m clock cycles.

However, since there are no strict requirements on the activation order of the secret shares, they can be activated in a random order at each secret computation, *e.g.*, generating a random permutation from the symmetric group S_m every m clock cycles. This acts as a shuffling countermeasure (within the masking scheme), which can be used to enhance the side-channel security of polynomial masking without penalties on the running time.

Algorithm 2 Modified FYD in Forward Direction.

```
Require: Input Permutation $\mathcal{P} = (p_1, ..., p_m)$
Ensure: Output Permutation $\mathcal{R} = (r_1, ..., r_m)$
 1: for $\omega = 1$ to $m$ do
 2:     $a_\omega \leftarrow p_\omega$
 3: end for
 4: for $\omega = 1$ to $m - 1$ do
 5:     $sel_\omega \overset{\$}{\leftarrow} [0, 2^{\lceil \log_2(m+1-\omega) \rceil} - 1]$
 6:     $j \leftarrow sel_\omega \bmod m + 1 - \omega$
 7:     $t \leftarrow a_\omega$
 8:     $a_\omega \leftarrow a_{j+\omega}$
 9:     $a_{j+\omega} \leftarrow t$
10: end for
11: for $\omega = 1$ to $m$ do
12:     $r_\omega \leftarrow a_\omega$
13: end for
```

The `shuffle` Module One simple way to create random permutations is to use the Fisher-Yates-Durstenfeld (FYD) algorithm, *e.g.*, in forward direction [44]. However, as the number of shares $m = 2d + 1$ is not a power of two, random permutations can be approximated using a slightly modified version of the FYD algorithm, as shown in Algorithm 2. This variant basically generates only the values $[0, m - \omega]$ for $\omega \in \{1, \ldots, m - 1\}$ using a modulo operation.

The `shuffle` module implements Algorithm 2 cascading a bunch of multiplexers, as illustrated in Figure 3.10.

The `shuffle` module requires $m - 1$ stages, in which $(m + 1 - \omega)$ multiplexers with $2^{\lceil \log_2(m+1-\omega) \rceil}$ inputs are deployed. In order to generate a random permutation, the select signals $\mathtt{sel}_\omega$ takes a uniform random value generated by a PRNG, which permutes the current input $(p_i)_{1 \le i \le m}$ into a new permutation $(r_i)_{1 \le i \le m}$ that gets stored into a register array consisting of $m \times \lceil \log_2(m) \rceil$-bit registers.

The `shuffle` module generates a new permutation (from the previous one) every m

clock cycles. Hence, two permutations are actually taking place during a polynomially masked multiplication. The new permutation is then used to shuffle the activation order of the shares within any other shared module during the next shared operation.

Note that the `shuffle` module can be used "as it is" to shuffle also the activation order of the shared key and state arrays within the AES.

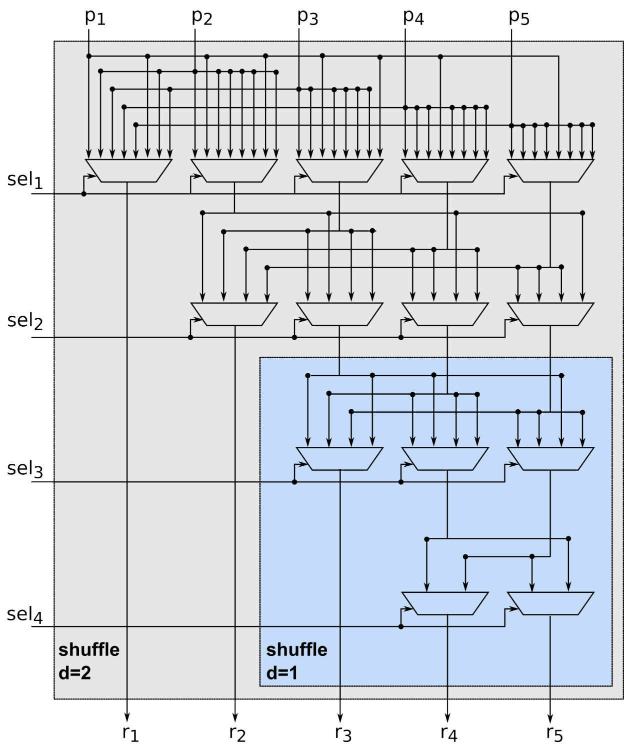


Figure 3.10: 1st-Order (blue box) and 2nd-Order (gray box) `shuffle` Module: These modules implement Algorithm 2 for $d = 1$ and $d = 2$, respectively. An output permutation $(r_1, ..., r_5)$ is generated from the input $(p_1, ..., p_5)$ according to the random values of the select signals $(\texttt{sel}_1, ..., \texttt{sel}_4)$.

3.5.2 Performance Evaluation

We synthesized the design using Synopsys Design Compiler J-2014.09-SP3 targeting a TSMC 45 nm standard technology library (`tcbn45gsbwptc`) at 1 MHz.

Our Gate Equivalent (GE) estimations were obtained using the smallest 2-to-1 NAND gate available in the library (`ND2D0BWP`), whose area accounts for 0.705 600 µm^2. We used the `compile` command *without* the `-ultra` parameter to allow for a fair comparison of the results with the literature. In this library, the GF(2^8) multiplier and adder cost 280 GE and 20 GE, respectively.

Table 3.1 summarizes the synthesis and performance results of our shuffled polynomially masked AES S-box implementation for $d \in [1, 6]$. The number of clock cycles and random bytes have been obtained from the formulas $35 \times m$ and $17 \times m \times d$, respectively.

Note that, while the number of clock cycles grows linearly in the masking order d, the number of random bytes as well as the area are $\mathcal{O}(d^2)$, being the size of the AES S-box dominated by the `shamul` module, whose area and randomness requirements grow quadratically in the masking order d. Also note that the maximum achievable clock frequency is expected to diminish with the masking order d, as the combinational paths get longer when increasing d.

Table 3.1: Performance Comparison of a Polynomially Masked AES S-box in $\mathsf{GF}(2^8)$ for Different Masking Orders.

	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
Speed [Clock Cycles]	105	175	245	315	385	455
Random [Bytes]	51	170	357	612	935	1326
Area [kGE]	10.2	33.4	79.7	157.8	276	442.7

3.6 Summary

In this chapter, we provided new principles for the selection of implementation-efficient addition chains, which lead to faster and more compact S-box implementations. These new principles were applied to the AES S-box, but they can be generally applied to any other S-box or power function.

Additionally, we proposed a lightweight shuffling countermeasure which randomly permutes the activation order of the shares within polynomial masking schemes, and that effectively increases their univariate security in practice as it will be shown in Chapter 5.

Note that this countermeasure applies to the masking modules themselves, and therefore it is agnostic of the underlying block-cipher. Furthermore note that, although the countermeasure was implemented in hardware, it can be analogously implemented in software too. Also, our shuffling scheme can be combined with further shuffling at higher algorithmic levels, *e.g.*, the execution order of the masked S-boxes can be further shuffled.

Based on our findings, we provided the design and synthesis results of a more efficient, shuffled and higher-order polynomial masked AES S-box, which randomly permutes the activation order of the shares within polynomial masking schemes.

In comparison to the previously known implementation of CHES 2013 [80], our 1st-order implementation requires 24% less DFFs, 21% less clock cycles, and 6% less randomness.

In particular, our 1st-order implementation requires 105 clock cycles and costs 10.2 kGE, while our 2nd-order implementation requires 175 clock cycles and costs 33.4 kGE when synthesized in ASIC.

Finally note that, although our implementation still requires quite some randomness, only 8-bit of randomness are needed per clock cycle and only every m clock cycles. This is fully within the capabilities of modern PRNGs.

4 Squeezing Polynomial Masking into Tower Fields

Contents

It is a very well known fact that subfield arithmetic can be used to reduce the size of cryptographic implementations defined over finite fields, being subfield elements smaller than the original field elements.

For the case of AES, this approach was presented by Vincent Rijmen in 2000 [98]. This seminal work showed how to compute the AES S-box, originally defined over the field $\mathsf{GF}(2^8)$, in the tower field $\mathsf{GF}((2^4)^2)$ through isomorphic transformations.

Since then, tower fields were employed many times in literature to reduce the area requirements of both unprotected and side-channel resistant hardware implementations of AES [102, 126, 82, 15], *i.e.*, small $\mathsf{GF}(2^4)$ arithmetic circuits were implemented in place of $\mathsf{GF}(2^8)$ circuits.

In this chapter, we take the tower field approach and present the design of a higher-order polynomially masked AES S-box in the tower field $\mathsf{GF}((2^4)^2)$, built around a small shared $\mathsf{GF}(2^4)$ multiplier.

Parts of this chapter have been published in the proceedings of the “Smart Card Research and Advanced Application Conference (CARDIS)” in 2016 [38].

Outline

- **Section 4.1** introduces tower fields arithmetic.
- **Section 4.2** provides a description of a polynomially masked tower field AES S-box.
- **Section 4.3** describes the hardware design of a polynomially masked tower field AES S-box and provides a comparison of the performance with state-of-the-art implementations.
- **Section 4.4** provides a summary of the chapter with a recapitulation of the main findings.

4.1 Arithmetic in Tower Fields

In this section, the basic arithmetic operations in the tower field $\mathsf{GF}((2^4)^2)$ are summarized. More details about AES implementations in tower fields can be found in [98, 126, 102].

Let $\mathsf{GF}((2^4)^2) \simeq \mathbb{F}_{2^4}[y]/(y^2+y+\gamma)$ denote the field of linear polynomials with coefficients in $\mathsf{GF}(2^4) \simeq \mathbb{F}_2[z]/(z^4+z+1)$. Let $\Phi : \mathsf{GF}(2^8) \to \mathsf{GF}((2^4)^2)$ denote the isomorphic mapping, which bijectively maps the elements from $\mathsf{GF}(2^8)$ to $\mathsf{GF}((2^4)^2)$, and $\Phi^{-1} : \mathsf{GF}((2^4)^2) \to \mathsf{GF}(2^8)$ its inverse.

Inversion The inversion $b = a^{-1}$ of an element $a \in \mathsf{GF}(2^8)$ can be computed in the tower field $\mathsf{GF}((2^4)^2)$ as:

$$\begin{cases} b_H &= a_H[a_H^2\gamma + a_H a_L + a_L^2]^{-1} \\ b_L &= (a_L + a_H)[a_H^2\gamma + a_H a_L + a_L^2]^{-1} \end{cases}, \tag{4.1}$$

where $a \mapsto_\Phi (a_H, a_L)$, $(b_H, b_L) \mapsto_{\Phi^{-1}} b$, and all operations are defined in $\mathsf{GF}(2^4)$.

Multiplication The multiplication $c = ab$ of two elements $a, b \in \mathsf{GF}(2^8)$ can be computed in $\mathsf{GF}((2^4)^2)$ as:

$$\begin{cases} c_H &= (b_H + b_L)(a_H + a_L) + b_L a_L \\ c_L &= b_H a_H \gamma + b_L a_L \end{cases}, \tag{4.2}$$

where $a \mapsto_\Phi (a_H, a_L)$, $b \mapsto_\Phi (b_H, b_L)$, $(c_H, c_L) \mapsto_{\Phi^{-1}} c$, and all operations are defined in $\mathsf{GF}(2^4)$.

Squaring The squaring $b = a^2$ of an element $a \in \mathsf{GF}(2^8)$ follows immediately from Equation (4.2):

$$\begin{cases} b_H &= a_H^2 \\ b_L &= a_H^2\gamma + a_L^2 \end{cases}, \tag{4.3}$$

where $a \mapsto_\Phi (a_H, a_L)$, $(b_H, b_L) \mapsto_{\Phi^{-1}} b$, and all operations are defined in $\mathsf{GF}(2^4)$.

4.2 Proposed Polynomially Masked AES S-box in $\mathsf{GF}((2^4)^2)$

The AES S-box $y = \mathsf{S}(x)$ in the tower field $\mathsf{GF}((2^4)^2) \simeq \mathbb{F}_{2^4}[y]/(y^2 + y + \texttt{0xE})$ can be written as follows:

$$\begin{cases} y_H &= \delta_{H,0} + \sum_{k=1}^{8} \delta_{H,k} z_H^{2^{k-1}} \\ y_L &= \delta_{L,0} + \sum_{k=1}^{8} \delta_{L,k} z_L^{2^{k-1}} \end{cases}, \tag{4.4}$$

where $\delta_H = (\delta_{H,k})_{0 \le k \le 8}$ are the coefficients defined by the vector

$$\delta_H = (\texttt{0xC}, \texttt{0x4}, \texttt{0x4}, \texttt{0x7}, \texttt{0x9}, \texttt{0x7}, \texttt{0x0}, \texttt{0x4}, \texttt{0xC}),$$

and $\delta_L = (\delta_{L,k})_{0 \le k \le 8}$ are the coefficients defined by the vector

$$\delta_L = (\texttt{0x7}, \texttt{0xB}, \texttt{0x1}, \texttt{0xC}, \texttt{0xA}, \texttt{0x7}, \texttt{0x1}, \texttt{0x7}, \texttt{0x9}),$$

$(z_H, z_L) = \Phi(x^{-1})$ is the result of inversion in $\mathsf{GF}((2^4)^2)$, the isomorphic mapping $(a_7, \ldots, a_0) \mapsto_\Phi (a_{H_3}, \ldots, a_{H_0}, a_{L_3}, \ldots, a_{L_0})$ is defined as

$$\begin{cases} a_{H_3} &= a_5 + a_7 \\ a_{H_2} &= a_2 + a_3 + a_5 + a_7 \\ a_{H_1} &= a_1 + a_4 + a_6 + a_7 \\ a_{H_0} &= a_4 + a_5 + a_6 \end{cases} \quad \begin{cases} a_{L_3} &= a_2 + a_4 \\ a_{L_2} &= a_1 + a_7 \\ a_{L_1} &= a_1 + a_2 \\ a_{L_0} &= a_0 + a_4 + a_5 + a_6 \end{cases}, \tag{4.5}$$

and the inverse mapping $(a_{H,3}, \ldots, a_{H,0}, a_{L,3}, \ldots, a_{L,0}) \mapsto_{\Phi^{-1}} (a_7, \ldots, a_0)$ is defined as

$$\begin{cases} a_7 &= a_{H_0} + a_{H_1} + a_{L_2} + a_{H_3} \\ a_6 &= a_{L_1} + a_{H_3} + a_{L_2} + a_{L_3} + a_{H_0} \\ a_5 &= a_{H_0} + a_{H_1} + a_{L_2} \\ a_4 &= a_{L_1} + a_{H_3} + a_{H_0} + a_{H_1} + a_{L_3} \end{cases} \quad \begin{cases} a_3 &= a_{H_0} + a_{H_1} + a_{L_1} + a_{H_2} \\ a_2 &= a_{H_0} + a_{H_1} + a_{L_1} + a_{H_3} \\ a_1 &= a_{H_0} + a_{H_1} + a_{H_3} \\ a_0 &= a_{L_0} + a_{H_0} \end{cases}. \tag{4.6}$$

The isomorphic mapping of the coefficients used in the AES S-box are summarized in Table 4.1.

Table 4.1: Isomorphic mapping $\Phi : \mathsf{GF}(2^8) \to \mathsf{GF}((2^4)^2)$ of the constants used in the AES S-box.

k	0	1	2	3	4	5	6	7	8
δ_k	0x63	0x05	0x09	0xF9	0x25	0xF4	0x01	0xB5	0x8F
$\Phi(\delta_k)$	0xC7	0x4B	0x41	0x7C	0x9A	0x77	0x01	0x47	0xC9

A Note on the Isomorphic Mappings All input values should be transformed using an isomorphic mapping $\Phi : \mathsf{GF}(2^8) \to \mathsf{GF}((2^4)^2)$ and, similarly, all the output values should be transformed back using the inverse isomorphic mapping $\Phi^{-1} : \mathsf{GF}((2^4)^2) \to \mathsf{GF}(2^8)$. As these transformations are defined over $\mathsf{GF}(2)$, they might be in principle problematic for polynomial masking schemes.

However, we suggest to apply them directly on the input/output values, *e.g.*, plaintexts, keys and ciphertexts, before/after each encryption, *i.e.*, before sharing and after reconstructing secrets. In this way, no special consideration for these mappings is required in the masking scheme.

4.3 Hardware Design

In this section, we present the design of a polynomially masked AES S-box in the tower field $\mathsf{GF}((2^4)^2)$ for any masking order $d = (m-1)/2$.

We denote $\mathsf{GF}(2^4) \simeq \mathbb{F}_2[z]/(z^4+z+1)$ polynomial basis multipliers by $\otimes$, while $\mathsf{GF}(2^4)$ adders are denoted by $\oplus$. The respective shared operations are denoted using the original symbols surrounded with boxes, *e.g.*, the shared $\mathsf{GF}(2^4)$ multiplier is denoted by $\boxed{\otimes}$.

The design basically consists of two hardware modules: a $\mathsf{GF}(2^4)$ shared multiplier (`shamul`) and a $\mathsf{GF}(2^4)$ shared multiplier by a constant equipped with a $\mathsf{GF}(2^4)$ shared adder (`shamac`).

We proceed as follows: first, we describe the design of the `shamul` and `shamac` modules for any masking order. Then, we describe how to use such modules to perform a shared inversion and a shared affine transformation in $\mathsf{GF}((2^4)^2)$ to securely compute the AES S-box at any order.

Finally, we present ASIC synthesis results up the 6^{th} masking order.

The `shamul` Module The `shamul` module implements the multiplication of two shared secrets in $\mathsf{GF}(2^4)$, according to Equation (3.3). In order to minimize the area requirements, it is used to perform both shared $\mathsf{GF}(2^4)$ multiplications and $\mathsf{GF}(2^4)$ squaring operations, as needed by the inversion and the affine transformation in $\mathsf{GF}((2^4)^2)$.

Similar to the shared multiplication module presented in Section 3.5, this `shamul` module instantiates $m^2 \times (d+1) + 2 \times m$ $\mathsf{GF}(2^4)$-multipliers, $m^2 \times (d+1) + m$ $\mathsf{GF}(2^4)$-adders, $m^2 + m \times d + 3 \times m$ 4-bit DFFs and a few multiplexers (MUXes). Note that all MUXes in the design are clocked using DFFs with enable in order to prevent glitches on the select lines [80].

Each shared multiplication requires a total of $4 \times m \times d$-bit randomness, whereas only $4 \times d$-bit are needed per clock cycle. To securely multiply two secrets it takes $2 \times m$ clock cycles in total. In the first m clock cycles, the input values are multiplied and re-masked according to (`Step 1`) and (`Step 2`) of Equation (3.3). In each clock cycle, only the i^{th} share is activated using the signals $(\mathtt{em_i})_{1 \le i \le m}$ and $(\mathtt{eout_i})_{1 \le i \le m}$.

This helps providing a sufficient separation of the leakages of each share over time. The signals $(\mathtt{selm_i})_{1 \le i \le m}$ and $(\mathtt{selout_i})_{1 \le i \le m}$ are used to select the inputs to the multiplication. In the subsequent m clock cycles, the result is reconstructed according to (`Step 3`) of Equation (3.3) activating one share at the time using the enable signals $(\mathtt{em_i})_{m+1 \le i \le 2m}$.

Finally, the results are stored in the appropriate registers using the $(\mathtt{el_i})_{1 \le i \le m}$ and $(\mathtt{eh_i})_{1 \le i \le m}$ signals for the low and high part, respectively. The low part is stored in $(\mathtt{el_i})_{1 \le i \le m}$ and the high part is moved from $(\mathtt{em_i})_{1 \le i \le m}$ to $(\mathtt{eh_i})_{1 \le i \le m}$ at the clock cycles $11 \times m + 1$ (1^{st} share is active) to $11 \times m + m$ (m^{th} share is active) (cf. Figure 4.2).

Note that the `shamul` module is additionally equipped with m $\mathsf{GF}(2^4)$-multipliers, $2 \times m$ $\mathsf{GF}(2^4)$-adders, and m 2-to-1 MUXes to perform also shared additions and shared multiplications by the constant `0xE`. This allows to compute shared multiplications and shared additions in parallel and save m clock cycles in the first step of the inversion and in the last step of the shared $\mathsf{GF}((2^4)^2)$ squaring.

The `shamac` Module The `shamac` module implements the shared multiplication by a constant and the shared addition of two shared secrets in $\mathsf{GF}((2^4)^2)$, as required by the affine transformation. Hence, the `shamac` module instantiates m circuits (shares), each one consisting of 3 $\mathsf{GF}(2^4)$-multipliers, 5 $\mathsf{GF}(2^4)$-adders, 5 4-bit DFFs, and a few MUXes, as illustrated in Figure 4.1.

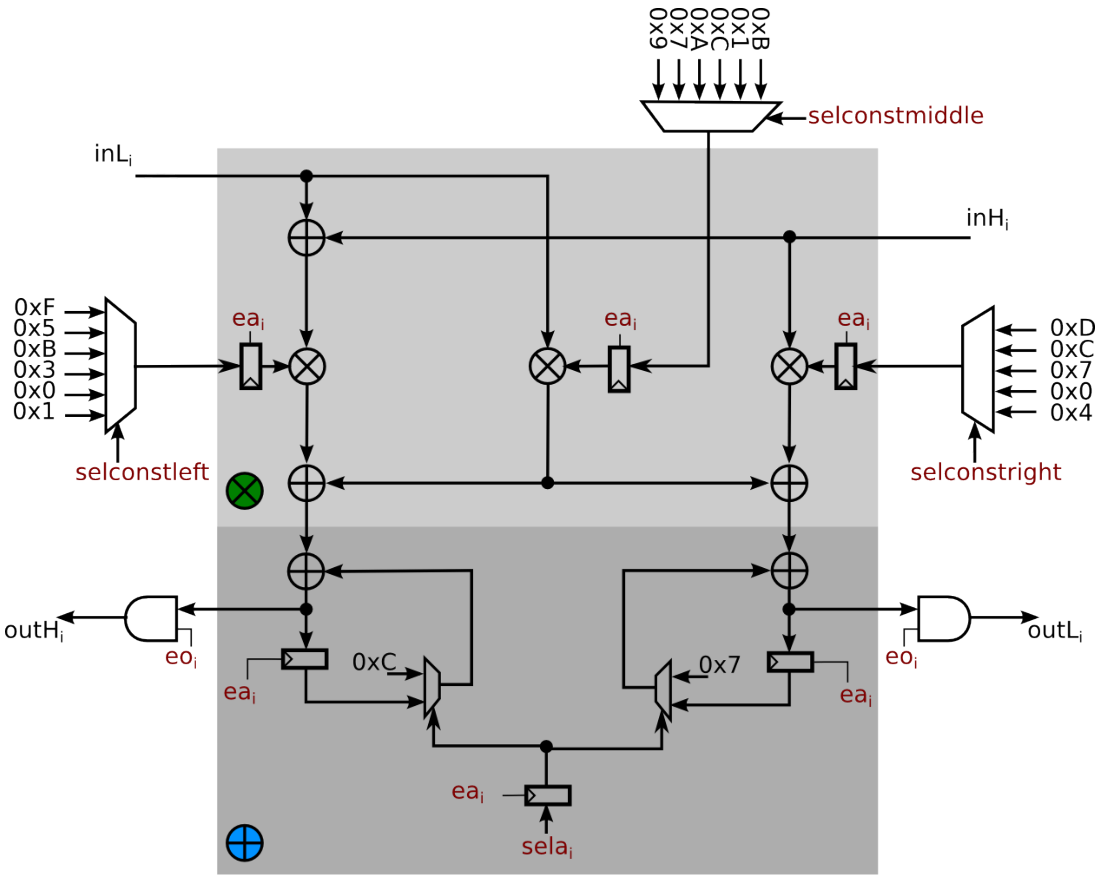


Figure 4.1: Multiplication and shared addition module in GF$((2^4)^2)$ for the i^{th} Share: This module must be instantiated m times and it is used as illustrated in Figure 4.3. The multiplication in GF$((2^4)^2)$ is performed according to Equation (4.2).

The `shamac` module takes m clock cycles and does not require fresh randomness: in the i^{th} clock cycle, only the i^{th} share is activated using the $(\mathtt{ea_i})_{1\leq i\leq m}$ signals, where the inputs are selected using the signals $(\mathtt{sela_i})_{1\leq i\leq m}$ and (`selconstleft`, `selconstmiddle`, `selconstright`).

The constant values, which are selected by the `selconst*` signals, are precomputed according to Equation (4.5) (cf. Table 4.1). The multiplication by a constant is performed according to Equation (4.2), where $(b_H, b_L) = \Phi(\delta_i)$.

More specifically, the signal (`selconstleft`) selects the constants obtained from the addition $\Phi(\delta_i)_H + \Phi(\delta_i)_L$, while the signal (`selconstright`) selects the constants obtained from the multiplication $\gamma\Phi(\delta_i)_H$. The values selected by (`selconstmiddle`) correspond to the values $\Phi(\delta_i)_L$. These pre-computations allow to save the area of 1

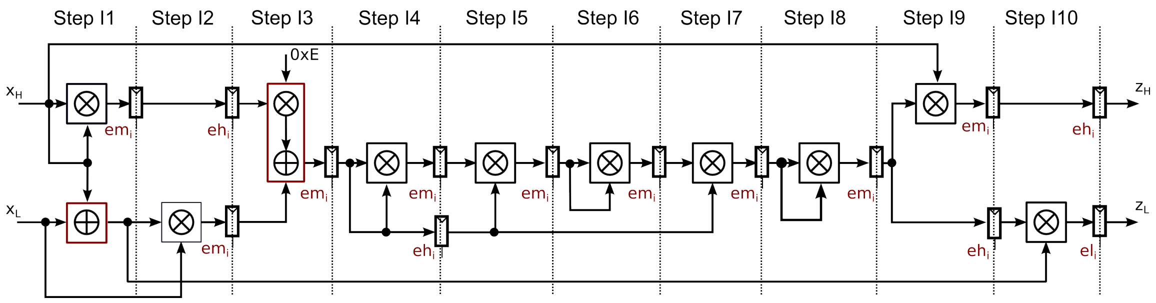


Figure 4.2: Evaluation of a polynomially masked inversion in $\mathsf{GF}((2^4)^2)$ using the addition chain $\mathcal{C}^{14}_{\mathsf{new}}$ inversion according to Equation (4.1).

$\mathsf{GF}(2^4)$-multiplier and 1 $\mathsf{GF}(2^4)$-adder per each share.

Finally, the $(\mathtt{eo_i})_{1 \leq i \leq m}$ signals are activated in the $54 \times m + 2$ clock cycle (1st share) to $55 \times m + 1$ clock cycle (m^{th} share) to output the result.

Note that the `shamac` module can be easily adapted to perform also the initial sharing, the round key additions, and the AES MixColumns operation.

All these operations require the shared multiplication by a constant and the shared addition of two shared secrets.

Shared Inversion The shared inversion in $\mathsf{GF}((2^4)^2)$ is computed according to Equation (4.1), where the inversion in $\mathsf{GF}(2^4)$ is computed using the addition chain $\mathcal{C}^{14}_{\mathsf{new}}$ of Figure 3.4.

The shared inversion is illustrated in Figure 4.2 (`[Step I1]` to `[Step I10]`) and consists of 9 shared multiplications, 1 shared addition, and 1 shared multiplication by a constant and addition.

In `[Step I1]`, a shared addition is performed in parallel to a shared multiplication, where the addition is performed during the first m clock cycles.

The output of the first shared multiplication is stored in the corresponding registers using the $\mathtt{em_i}$ signals. The result of the shared addition must be selected for `[Step I2]` and `[Step I10]` using the $\mathtt{selm_i}$ signal. The output of shared multiplication in `[Step I2]` is then stored in the $\mathtt{em_i}$ registers and the values of `[Step I1]` are moved to the $\mathtt{eh_i}$ registers.

This means that in each clock cycle $1 \leq i \leq m$ of `[Step I3]`, `[Step I5]` and `[Step I10]` the signal $\mathtt{eh_i}$ is enabled, in order to move the content of the registers $\mathtt{em_i}$ from the previous steps into the $\mathtt{eh_i}$ registers. For `[Step I3]`, the output of the shared

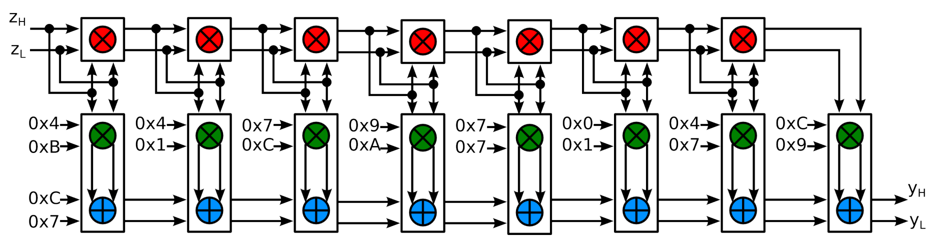


Figure 4.3: Evaluation of a polynomially masked affine transformation in $\mathsf{GF}((2^4)^2)$ using the addition chain $\mathcal{C}^{14}_{\mathsf{new}}$ inversion according to Equation (4.4).

multiplication with the constant `0xE` is selected using the `selout`$_\mathtt{i}$ signal, which is enabled over the signal `eout`$_\mathtt{i}$. Therefore only m clocks cycles are needed and the results are stored into the `em`$_\mathtt{i}$ registers.

After `[Step I3]`, the shared addition chain $\mathcal{C}_{14}$ is calculated in `[Step I4]` to `[Step I8]` and stored in the `em`$_\mathtt{i}$ registers. Therefore, the outputs of `[Step I3]` are stored in the `eh`$_\mathtt{i}$ registers, for `[Step I7]`. In the `[Step I9]` and `[Step I10]`, the last two shared multiplications are calculated and the output values are stored in the `eh`$_\mathtt{i}$ and `el`$_\mathtt{i}$ register to perform the affine transformation.

Overall the shared inversion takes $19 \times m = 38 \times d + 19$ clock cycles and requires $36 \times m \times d = 72 \times d^2 + 36 \times d$-bit of randomness to compute the inverse in the tower field $\mathsf{GF}((2^4)^2)$.

Shared Affine Transformation The affine transformation consists of 7 squaring operations and 8 multiplications by a constant (cf. Figure 4.3). The 7 squarings and the first 7 multiplications with the constants run in parallel.

Each squaring operation consists of 2 shared $\mathsf{GF}(2^4)$ multiplications (squarings), 1 shared $\mathsf{GF}(2^4)$ multiplication by a constant and a final $\mathsf{GF}(2^4)$ addition (cf. Equation (4.3)). The operations needed for one shared squaring operation in $\mathsf{GF}((2^4)^2)$ are illustrated in Figure 4.4 and consist of three steps (`[Step A1]` to `[Step A3]`).

The shared multiplication in `[Step A1]` uses the `el`$_\mathtt{i}$ register output after `[Step I10]` for the first squaring in $\mathsf{GF}((2^4)^2)$ and the `el`$_\mathtt{i}$ register output after `[Step A3]` for the following ones. The computation of `[Step A2]` and `[Step A3]` are performed in the same way as in `[Step I2]` and `[Step I3]`. Hence, the multipliers and addition modules from `shamul` can be reused.

In total, the shared affine transformation requires $36 \times m = 72 \times d + 36$ clock cycles and $56 \times m \times d = 112 \times d^2 + 56 \times d$-bit of randomness, where one squaring costs $5 \times m = 10 \times d + 5$ clock cycles and uses $8 \times m \times d = 16 \times d^2 + 8$-bit of randomness.

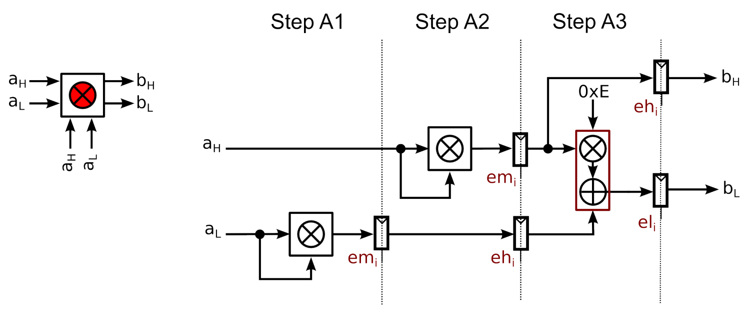


Figure 4.4: Evaluation of a polynomially masked squaring operation in $\mathsf{GF}((2^4)^2)$ according to Equation (4.3).

4.3.1 Performance Comparison

We synthesized our design using Synopsys Design Compiler J-2014.09-SP3 and targeting a TSMC 45 nm standard technology library (`tcbn45gsbwptc`) at 1 MHz.

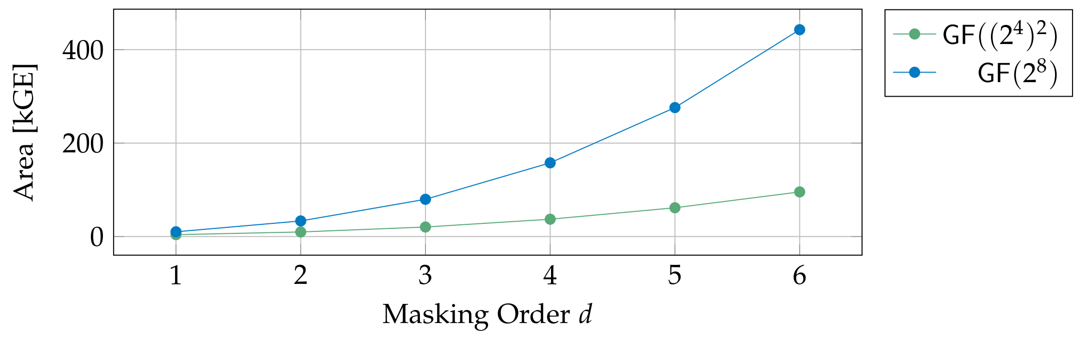
As done in Section 3.5.2, our Gate Equivalent (GE) estimations were obtained using the smallest 2-to-1 NAND gate available in the library (`ND2D0BWP`), whose area accounts for 0.7056 µm^2, and the `compile` command to enable a fair comparison of the results.

The $\mathsf{GF}(2^4)$ multiplier costs only 53 GE, while the $\mathsf{GF}(2^4)$ adder costs 10 GE in our library. In total, the computation of a shared AES S-box requires $110 \times d + 55$ clock cycles and $184 \times d^2 + 92 \times d$-bit of fresh randomness.

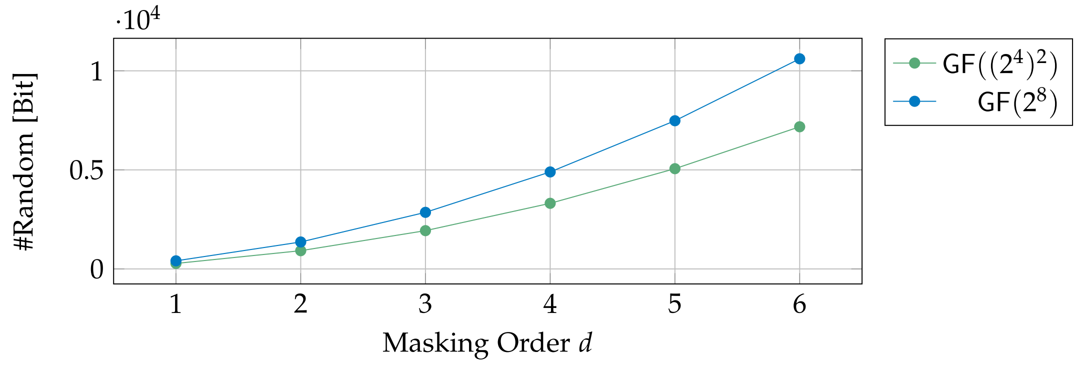
Table 4.2 provides an overview on the performance of our design in $\mathsf{GF}((2^4)^2)$. In Figure 4.5, these results are further compared to the improved $\mathsf{GF}(2^8)$ design described in Chapter 3.

Table 4.2: Performance Comparison of a Polynomially Masked AES S-box in $\mathsf{GF}((2^4)^2)$ for Different Masking Orders.

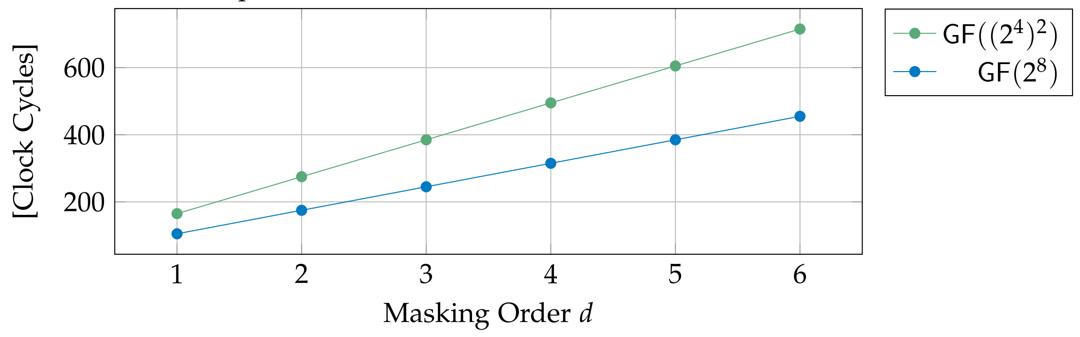
$\mathsf{GF}((2^4)^2)$	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
Area [kGE]	4.0	9.8	20.4	37.1	61.6	95.5
Speed [Clock Cycles]	165	275	385	495	605	715
Random [Bit]	276	920	1,932	3,312	5,060	7,176



(a) Performance comparison for area size in kGE.



(b) Performance comparison for randomness in number of bits.



(c) Performance comparison for speed in number of clock cycles.

Figure 4.5: Performance Comparison of a Polynomially Masked AES S-box in GF(2^8) vs. GF$((2^4)^2)$ for Different Masking Orders $d \in \{1, 2, 3, 4, 5, 6\}$.

4.4 Summary

In this section, we showed how to reduce the area and randomness requirements of higher-order polynomial masking hardware implementations using tower fields. This approach was applied to the AES S-box, but it can be similarly applied to any other S-box which can be defined over tower fields.

Our 1st-order and 2nd-order masked implementations of the AES S-box in $\mathsf{GF}((2^4)^2)$ cost 4.0 kGE and 9.8 kGE, respectively. These results highlight that our 1st-order implementation over tower fields requires 60% less area the corresponding 1st-order implementation in the original AES field and the 2nd-order implementation over tower fields is as big as the 1st-order implementation in the original AES field.

The area size of the 1st-order implementation over tower fields is also comparable to those obtained by initial TI implementations of the AES S-box [82, 15] and to the size of a polynomially masked PRESENT S-box [36].

Finally, the number of random bits results also drastically reduced and the randomness requirements per clock cycle have been further reduced down to only 4-bit per clock cycle (every m clock cycles).

5 Electro-Magnetic Field Analysis of Polynomial Masking Schemes

Contents

EM field side-channel analysis is particularly relevant in those cases where modifying the target device to insert a shunt resistor might permanently damage it, or it would require too long time and expertise, *e.g.*, due to inaccessible contacts in multi-layers PCB boards or ICs with multiple power and ground pins connected with other components on the board.

In this chapter, we provide a side-channel analysis evaluation of previously presented polynomially masked implementations using EM analysis and side-channel correlation-collision attacks [81].

Side-channel correlation-collision attacks have been shown to be practically effective against polynomially masked FPGA implementations in previous works (cf. [80]). This kind of attacks has the advantage that they not require either the knowledge of a precise power model nor to own a known-key device for profiling.

Parts of this chapter have been published at the "Theory of Implementation Security Workshop (TIs)" in 2016 [37] and in the proceedings of the "Smart Card Research and Advanced Application Conference (CARDIS)" in 2016 [38].

Outline

- **Section 5.1** introduces our evaluation methodology including a description of the used measurement setup and attacks performed.
- **Section 5.2** presents the results of our side-channel analysis investigations.
- **Section 5.3** provides a summary of the chapter with a recapitulation of the main findings.

5.1 Evaluation Methodology

5.1.1 Measurement Setup

We synthesized the design using Xilinx ISE Webpack version 14.7 targeting a Xilinx Spartan-6 LX9 FPGA in FTG256 Ball Grid Array (BGA) package (`xc6slx9-2-ftg256`). The device is implemented in a 45 nm process technology and uses a 1.2 V power supply for the internal core logic.

We used an Agilent DSO9254A 2.5 GHz digital oscilloscope with a 10 mm diameter magnetic field probe (`RF-R 50-1` from Langer EMV-Technik[1]) connected to a 30 dB pre-amplifier (`PA303` from Langer EMV-Technik) to fully exploit the vertical resolution of the oscilloscope.

Differently from [80], we did not employ any Direct Current (DC) blocker, as our EM probe is active in the frequency range 30 MHz$-$3 GHz.

The oscilloscope sampled at 100 MSa/s and we clocked our design at 4 MHz, resulting in 25 Sa/cycle for all our experiments (where not otherwise specified). Note that we did not observe any further leakages when sampling at higher sampling rates.

We focused our analysis on the first two S-box operations and applied 1^{st}- and 2^{nd}-order side-channel correlation-collision attacks [81], while fixing the security level at 10, 000, 000 measurements (where not otherwise specified).

For the sake of evaluation, we fixed the public points. Note that, this is a worst-case assumption, as the public points are not required to be fixed in practice, hence reducing the effectiveness of our attacks in a real scenario.

Assuming that the points are uniformly and independently drawn at random, the probability that the adversary can observe the same public points is $\prod_{i=0}^{m-1} \frac{1}{(2^n-1-i)}$,

[1] `https://www.langer-emv.de`

where n specify the size of the extension field $\mathsf{GF}(2^n)$.

This means that it would be much harder to mount successful attacks in practice, especially for $d > 1$, as the probability decreases quickly with the masking order d.

5.1.2 Side-Channel Collision Attacks

A function f has collisions if it has the same output for different inputs.

In the context of side-channel analysis, the input is typically the output of a function g of some public data m and some secret data k, *e.g.*, $f(g(m,k))$. Hence, f can produce collisions even if it is bijective by a careful choice of the inputs m and k.

Since collisions are key-dependent, then they can be exploited to distinguish intermediate values during cryptographic computations. Note that side-channel collision attacks require no previous knowledge on the distribution of the leakage information.

Side-channel collision attacks were originally introduced in [110, 109] and further improved in [81] under the name of correlation-enhanced collision attacks. We use this latter variant for our investigations.

Attack description By the way of an example, let consider the leakages associated to the look-up of two bijective b-bit S-boxes S after an initial key addition, like in the case of AES. Then, S produces collisions if:

$$\mathsf{S}(m_1 \oplus k_1) = \mathsf{S}(m_2 \oplus k_2) \quad \Longleftrightarrow \quad m_1 \oplus m_2 = k_1 \oplus k_2 = \Delta_k,$$

where m_i and k_i correspond to the i^{th} byte of the plaintext and key, respectively.

Since S is a bijective function, then there is a collision for every input given two fixed keys (k_1, k_2):

$$\forall m_1\ \exists! m_2 \colon\ \mathsf{S}(m_1 \oplus k_1) = \mathsf{S}(m_2 \oplus k_2)$$

The main idea of the attack is to classify leakages for different key hypothesis Δ_k such that the classification made for the correct key difference is distinguishable from the classifications made for wrong key hypotheses.

Since there is a collision for every input given a fixed key difference, then it is sufficient to test all possible key distances for every input to find out the correct key.

In particular, for every key hypothesis Δ_k a pair of sets (S_1, S_2) is created for every possible input $m \in \{0, ..., 2^b - 1\}$ and the leakages of the two S-boxes are classified into the two sets S_1 and S_2, respectively.

The correlation-based distinguisher calculates the sample mean of each pair set $(\overline{S}_1^m, \overline{S}_2^m)$ and then computes the correlation coefficient ρ_{Δ_k} for every key hypothesis Δ_k as follows:

$$\rho_{\Delta_k}\left([\overline{S}_1^0, \ldots, \overline{S}_1^{(2^b-1)}], [\overline{S}_2^{0\oplus\Delta_k}, \ldots, \overline{S}_2^{(2^b-1)\oplus\Delta_k}]\right)$$

5.2 Experimental Results on FPGA

For our side-channel investigations, we considered the implementations presented in Chapter 3 and Chapter 4 and proceeded as follows:

1. we considered a 1st-order implementation and verified the existence of 1st-order univariate leakage, when the masks were off;
2. we activated the PRNG to generate the masks and confirmed the existence of 2nd-order univariate leakage in the EM domain;
3. we activated the shuffling countermeasure to verify its effectiveness to hide the 2nd-order univariate leakage in 1st-order implementations;
4. we verified the univariate higher-order security of a 2nd-order implementation without shuffling.

We conclude that our 1st-order implementation with shuffling and our 2nd-order implementation achieve univariate 1st- and 2nd-order security up to $10,000,000$ traces in a worst-case scenario analysis.

5.2.1 $\mathsf{GF}(2^8)$ AES S-box Implementation

In this section, the design presented in Section 3.5 is considered.

The public points $(\alpha_i)_{1\leq i\leq m}$ and $(\lambda_i)_{1\leq i\leq m}$ are fixed to (`0x02`, `0x03`, `0x04`) and (`0x02`, `0xD2`, `0xD1`) for $d = 1$ and to (`0x04`, `0x05`, `0x06`, `0x0A`, `0x0B`) and (`0x64`, `0x94`, `0xAE`, `0x9F`, `0xC0`) for $d = 2$, respectively.

1st-Order Masking and PRNG Off As a sanity check, we verified the existence of 1st-order univariate leakage in our 1st-order implementation, when all masks were set to zero.

The results are reported in Figure 5.1, which clearly shows how the EM activity of two S-box evaluations perfectly correlates over time, when the masks are off.

This analysis was conducted with only $1,000,000$ measurements.

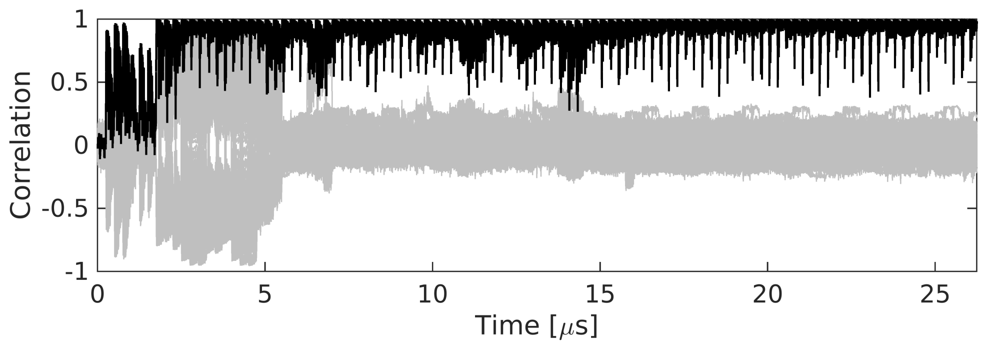


Figure 5.1: 1st-Order $\mathsf{GF}(2^8)$ Masked Implementation, PRNG Off, Shuffling Off, 1st-Order Side-Channel Correlation Collision Univariate Attacks, $1,000,000$ Traces, $f_{\text{clk}} = 4\,\text{MHz}$. The correlation coefficient for the correct key is marked in **black**, while **gray** is used to depict wrong key hypotheses.

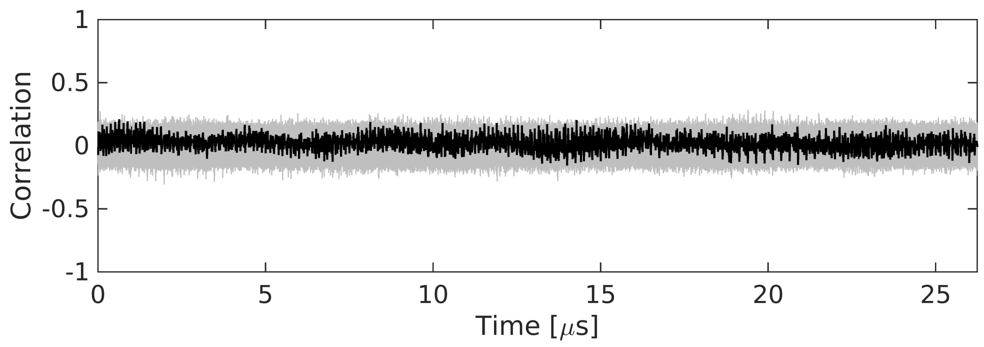
1st-Order Masking, PRNG On and Shuffling Off The results of the analysis of the 1st- and 2nd-order univariate leakage, when the shuffling countermeasure is deactivated, is reported in Figure 5.2.

Even though the `shamul` and `shamac` module are active at the same time, no leakage can be observed in the 1st-order moment.

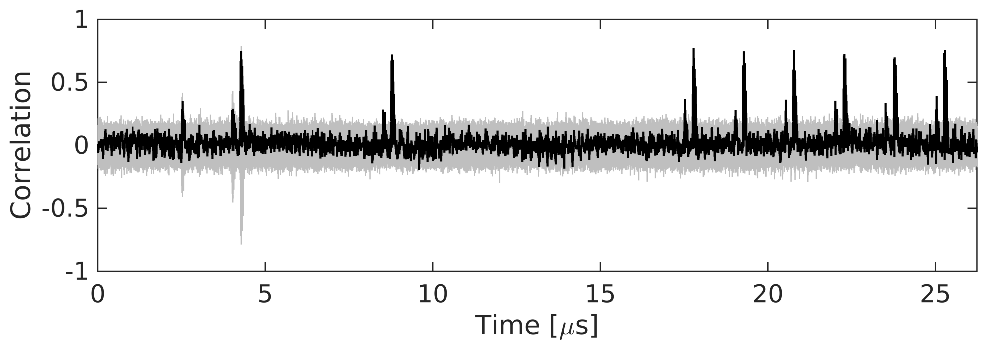
However, a significant leakage can be observed in the 2nd-order moment during both the inversion and the affine transformation.

We repeated the same attacks by clocking the target device with increasing clock frequencies.

The results are illustrated in Figure 5.3 and show that the efficiency of 2nd-order attacks increases with the clock frequency, confirming the results obtained in [80] also in the EM domain.



(a) 1st-Order Side-Channel Correlation Collision Univariate Attacks: No leakage can be observed over the S-box computation.



(b) 2nd-Order Side-Channel Correlation Collision Univariate Attacks: leakage can be observed during both inversion and the affine transformation.

Figure 5.2: 1st-Order $\mathsf{GF}(2^8)$ Masked Implementation, PRNG On, Shuffling Off, 1st-Order (top) and 2nd-Order Side-Channel Correlation Collision Univariate Attacks (bottom), 10, 000, 000 Traces, $f_{\text{clk}} = 4\,\text{MHz}$. The correlation coefficient for the correct key is marked in **black**, while gray is used to depict wrong key hypotheses.

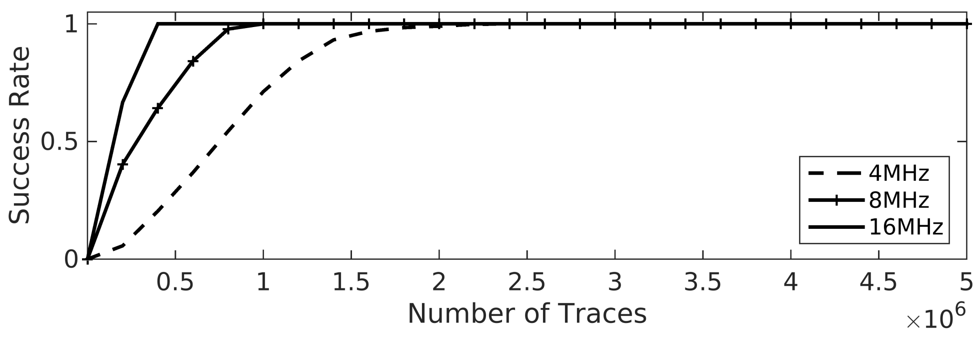
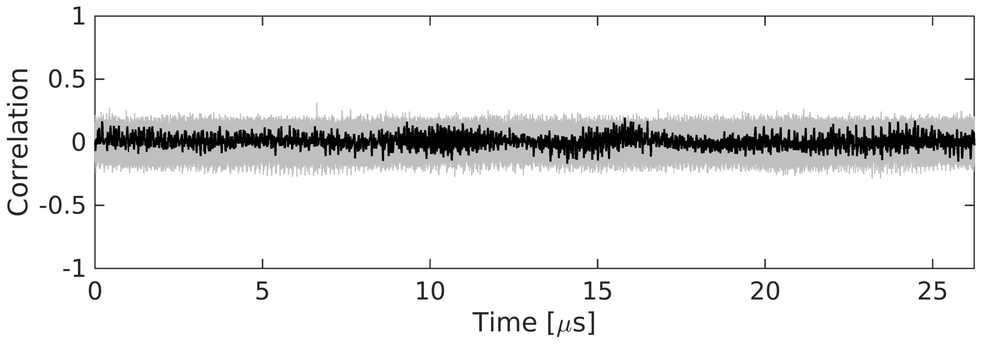
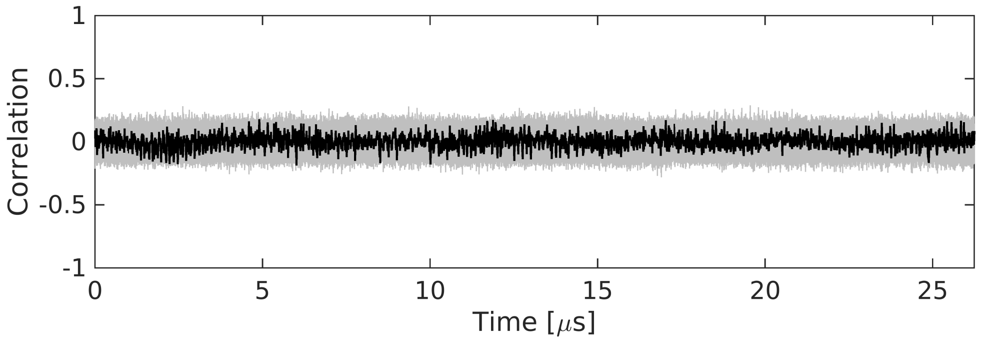


Figure 5.3: 1st-Order $\mathsf{GF}(2^8)$ Masked Implementation, PRNG On, Shuffling Off, 2nd-Order Side-Channel Correlation Collision Univariate Attacks: Success rate for increasing clock frequencies showing that the efficiency of 2nd-order attacks increases with the clock frequency.

1st-Order Masking, PRNG On, and Shuffling On The results of the analysis of the 1st- and 2nd-order univariate leakage, when the shuffling countermeasure is on, are reported in Figure 5.4 and Figure 5.5. Interestingly, no univariate leakage can be observed in any of the first two moments at the considered security level. Also, MIA does not reveal any other leakage sources in the traces.



(a) 1st-Order Side-Channel Correlation Collision Univariate Attacks.



(b) 2nd-Order Side-Channel Correlation Collision Univariate Attacks.

Figure 5.4: 1st-Order $\mathsf{GF}(2^8)$ Masked Implementation, PRNG On, Shuffling On, 1st-Order (top) and 2nd-Order Side-Channel Correlation Collision Univariate Attacks (bottom), 10, 000, 000 Traces, $f_{\text{clk}} = 4\,\text{MHz}$. The correlation coefficient for the correct key is marked in **black**, while **gray** is used to depict wrong key hypotheses.

2nd-Order Masking, PRNG On and Shuffling Off Finally, the results of the analysis of the 1st- and 2nd-order univariate leakage, when the shuffling countermeasure is off, for a 2nd-order implementation are reported in Figure 5.6 and Figure 5.7. As for the previous case, it is not possible to observe any leakages in the 1st and 2nd order, but also no other leakages are detected by either higher-moments (up to the third) nor using mutual information, for the considered security level.

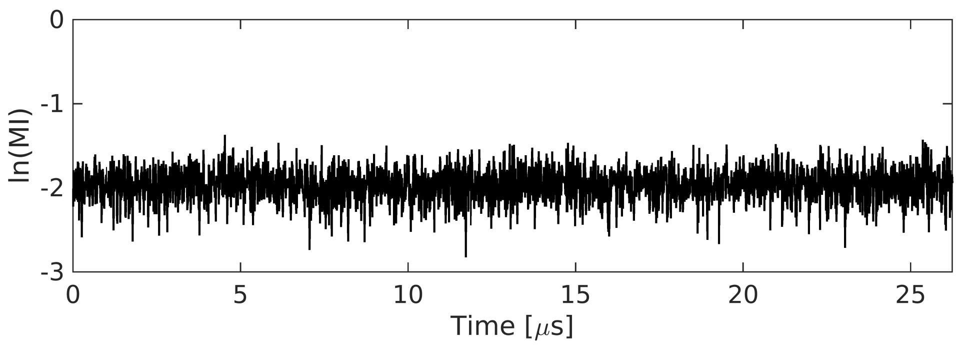
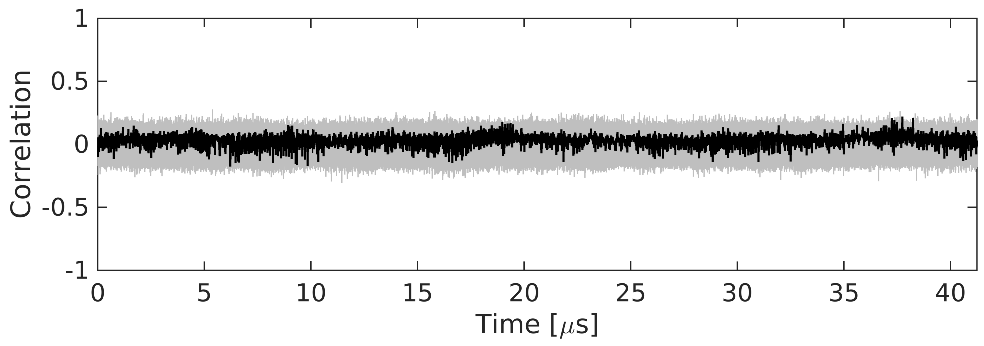
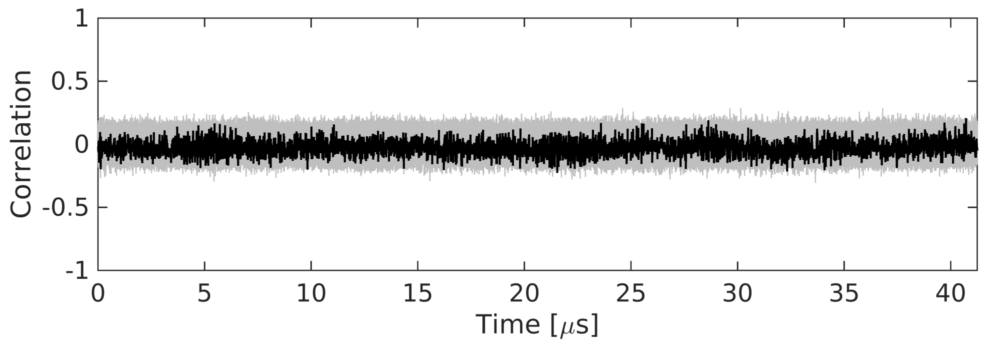


Figure 5.5: 1st-Order $\mathsf{GF}(2^8)$ Masked Implementation, PRNG On, Shuffling On, Mutual Information estimated using Histograms, $10,000,000$ Traces, $f_{\text{clk}} = 4\,\text{MHz}$.

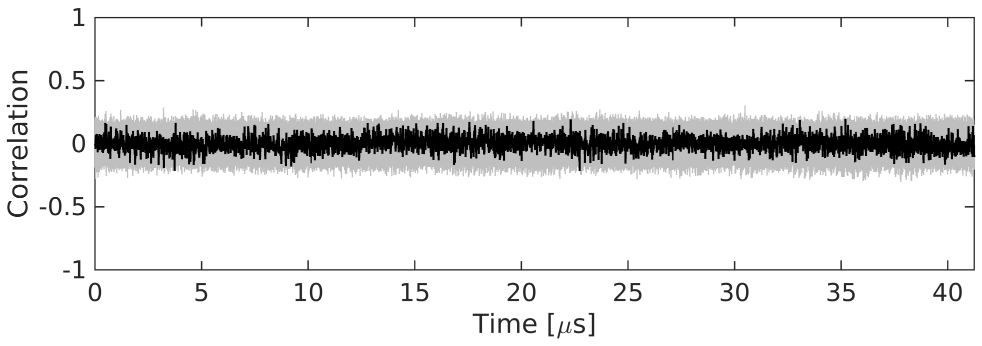


(a) 1st-Order Side-Channel Correlation Collision Univariate Attacks.

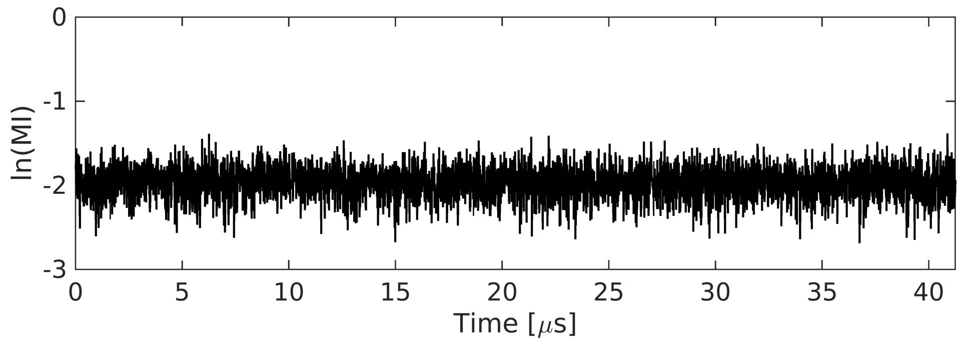


(b) 2nd-Order Side-Channel Correlation Collision Univariate Attacks.

Figure 5.6: 2nd-Order $\mathsf{GF}(2^8)$ Masked Implementation, PRNG On, Shuffling Off, 1st-Order (top) and 2nd-Order Side-Channel Correlation Collision Univariate Attacks (bottom), $10,000,000$ Traces, $f_{\text{clk}} = 4\,\text{MHz}$. The correlation coefficient for the correct key is marked in **black**, while gray is used to depict wrong key hypotheses.



(a) 3rd-Order Side-Channel Correlation Collision Univariate Attacks. The correlation coefficient for the correct key is marked in **black**, while **gray** is used to depict wrong key hypotheses.



(b) Mutual Information estimated using Histograms.

Figure 5.7: 2nd-Order $\mathsf{GF}(2^8)$ Masked Implementation, PRNG On, Shuffling Off, 3^{rd}-Order Side-Channel Correlation Collision Univariate Attacks (top) and Mutual Information (bottom), $10,000,000$ Traces, $f_{\text{clk}} = 4\,\text{MHz}$.

5.2.2 $\mathsf{GF}((2^4)^2)$ AES S-box Implementation

The results of the side-channel analysis of the design introduced in Section 4.3 are presented. The public points $(\alpha_i)_{1\le i\le m}$ and $(\lambda_i)_{1\le i\le m}$ are fixed to $(\texttt{0x2},\texttt{0x3},\texttt{0x4})$ and $(\texttt{0x2},\texttt{0x5},\texttt{0x6})$ for $d = 1$ and to $(\texttt{0x4},\texttt{0x5},\texttt{0x6},\texttt{0x0A},\texttt{0x0B})$ and $(\texttt{0x6},\texttt{0xE},\texttt{0x4},\texttt{0x5},\texttt{0x8})$ for $d = 2$, respectively.

1st-Order Masking and Masks Off We verified the existence of 1st-order univariate leakages, when the masks are off, using only $100,000$ measurements. The results are reported in Figure 5.8, clearly showing the existence of leakages during the whole S-box computations.

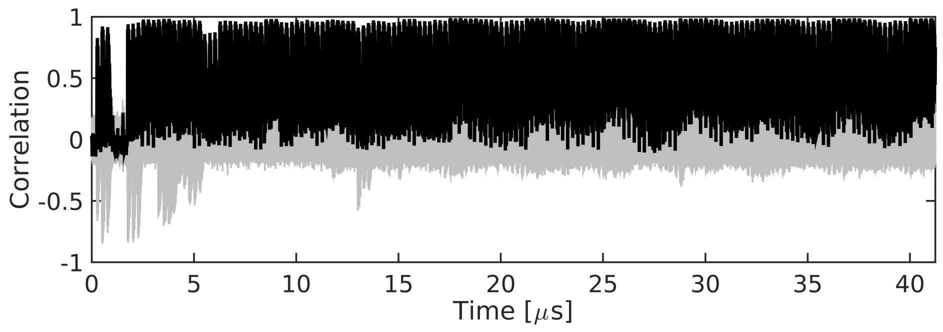


Figure 5.8: 1st-Order $\mathsf{GF}((2^4)^2)$ Masked Implementation, Masks Off, 1st-Order Side-Channel Correlation Collision Univariate Attacks, 100,000 Traces, $f_{\mathrm{clk}} = 4\,\mathrm{MHz}$. The correlation coefficient for the correct key is marked in **black**, while gray is used to depict wrong key hypotheses.

1st-Order Masking, Masks On and Shuffling Off The results of 1st-order and 2nd-order univariate attacks are shown in Figure 5.9 and Figure 5.10. Even though the `shamul` and `shamac` modules are active at the same time, no leakage can be observed in the 1st-order. However, similarly to the previous analysis, leakage information is present in the 2nd-order during the affine transformation.

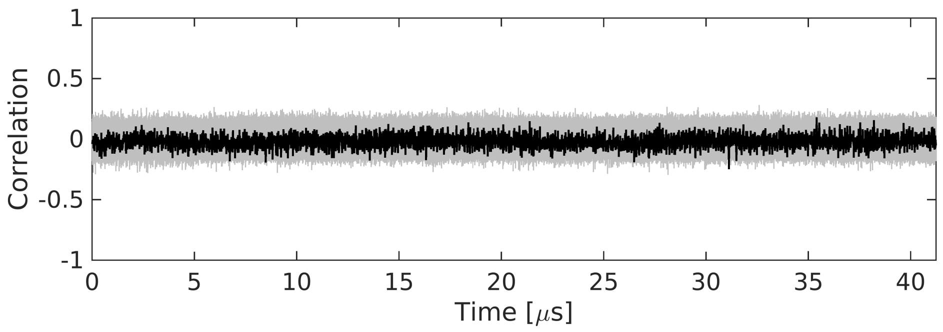


Figure 5.9: 1st-Order $\mathsf{GF}((2^4)^2)$ Masked Implementation, Masks On, Shuffling Off, 1st-Order Side-Channel Correlation Collision Univariate Attacks, 10,000,000 Traces, $f_{\mathrm{clk}} = 4\,\mathrm{MHz}$. The correlation coefficient for the correct key is marked in **black**, while gray is used to depict wrong key hypotheses.

1st-Order Masking, Masks On and Shuffling On The results of the analysis is reported in Figure 5.11, when the activation order of the secret shares is shuffled using the countermeasure described in Section 3.5.1. Interestingly, also mutual information analysis does not reveal the presence of any leakages (cf. Figure 5.12).

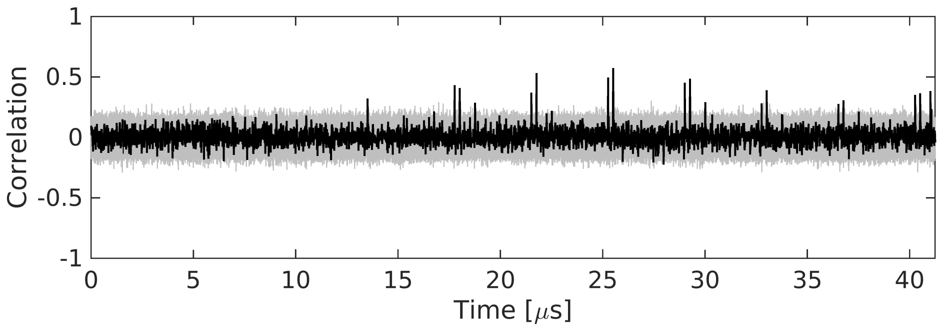
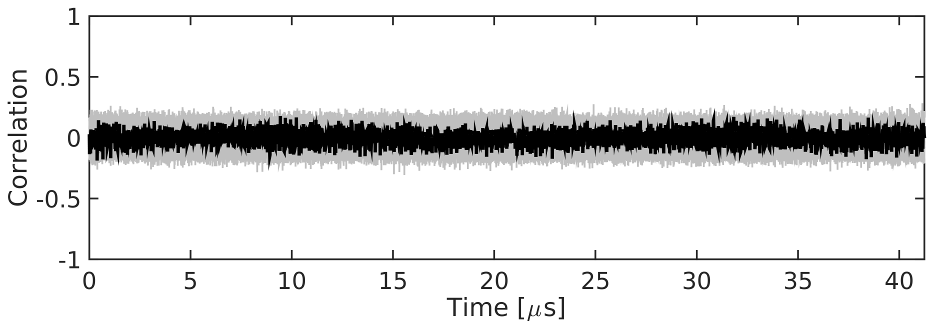
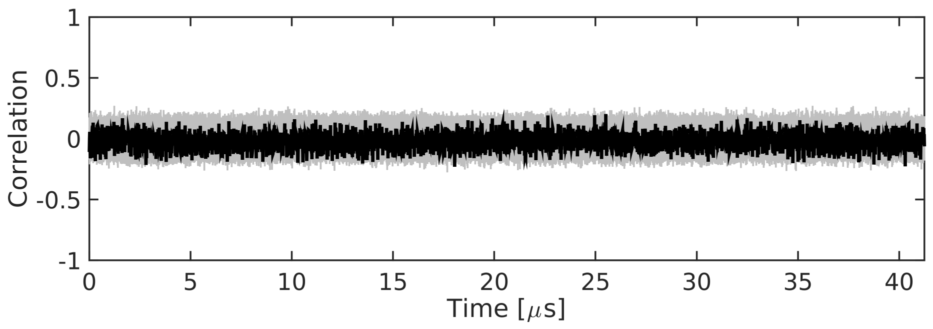


Figure 5.10: 1st-Order $\mathsf{GF}((2^4)^2)$ Masked Implementation, Masks On, Shuffling Off, 2nd-Order Side-Channel Correlation Collision Univariate Attacks, 10,000,000 Traces, $f_{\text{clk}} = 4\,\text{MHz}$. The correlation coefficient for the correct key is marked in **black**, while gray is used to depict wrong key hypotheses.



(a) 1st-Order Side-Channel Correlation Collision Univariate Attacks.



(b) 2nd-Order Side-Channel Correlation Collision Univariate Attacks.

Figure 5.11: 1st-Order $\mathsf{GF}((2^4)^2)$ Masked Implementation, Masks On, Shuffling On, 1st-Order (top) and 2nd-Order Side-Channel Correlation Collision Univariate Attacks (bottom), 10,000,000 Traces, $f_{\text{clk}} = 4\,\text{MHz}$. The correlation coefficient for the correct key is marked in **black**, while gray is used to depict wrong key hypotheses.

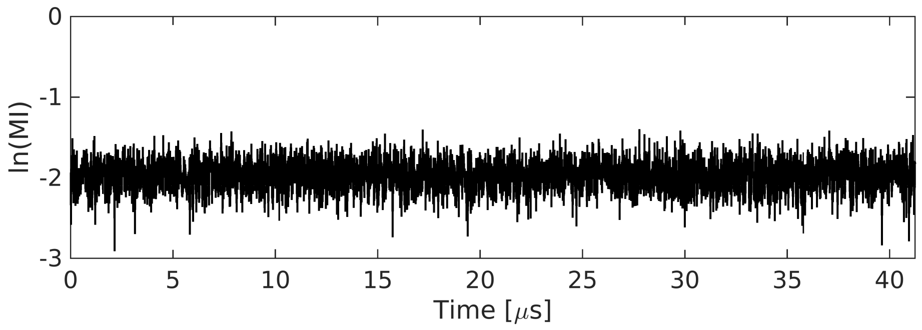


Figure 5.12: 1st-Order $\mathsf{GF}((2^4)^2)$ Masked Implementation, Masks On, Shuffling On, Mutual Information estimated using Histograms, 10,000,000 Traces, $f_{\text{clk}} = 4\,\text{MHz}$.

2nd-Order Masking, Masks On and Shuffling Off The results of 1st-order and 2nd-order univariate side-channel attacks against a 2nd-order implementation without shuffling are reported in Figure 5.13 and Figure 5.14. Also in this case, no univariate leakages can be observed either in any of the first third moments nor using mutual information analysis.

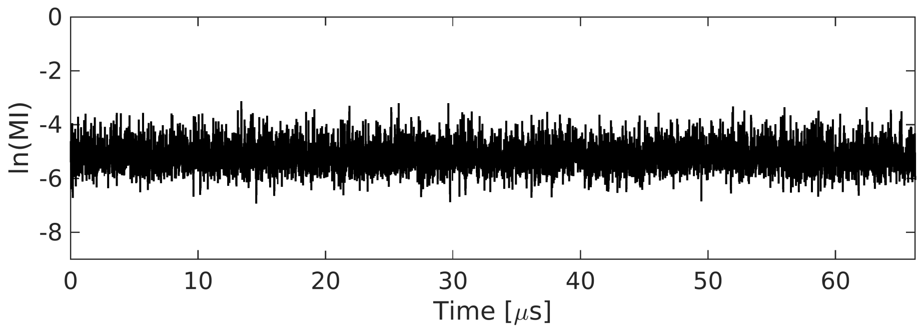


Figure 5.13: 2nd-Order $\mathsf{GF}((2^4)^2)$ Masked Implementation, Masks On, Shuffling Off, Mutual Information estimated using Histograms, 10,000,000 Traces, $f_{\text{clk}} = 4\,\text{MHz}$.

5.3 Summary

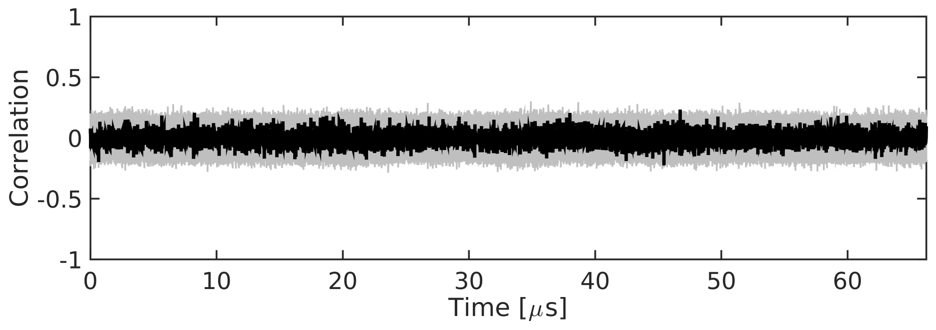
In this chapter, we provided the results of a side-channel analysis evaluation of the polynomially masked implementations presented in Chapter 3 and Chapter 4 using EM analysis and side-channel correlation-collision attacks.

In particular, we showed the effectiveness of our proposed lightweight shuffling countermeasure, which randomly permutes the activation order of the shares within polynomial masking schemes, against higher-order univariate side-channel collision attacks.

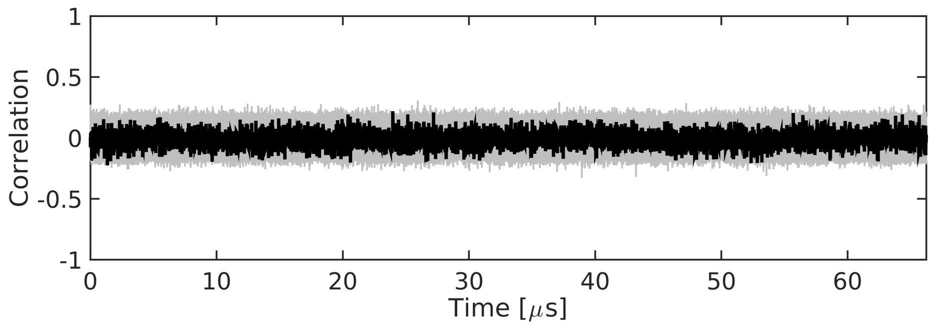
Our experiments showed that higher-order univariate security is guaranteed up to $10,000,000$ traces in a worst-case scenario analysis using fixed public points, thus providing a significant security improvement at negligible area costs.

It is worth noting that shuffling the activation order of the shares seems to be particularly effective against side-channel collision attacks, as each operation within the scheme is independently shuffled, thus making collisions harder to detect in practice.

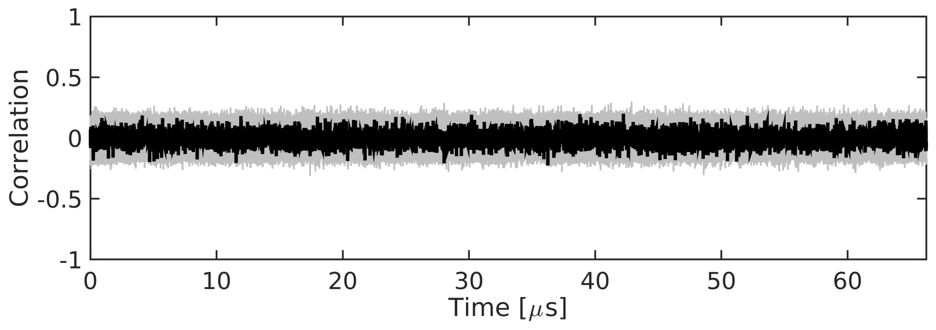
However, it can also be noted that the leakages are only hidden by the shuffling countermeasures, thus some leakage is expected to emerge either by increasing the number of measurements or by using more sophisticated multivariate attacks.



(a) 1st-Order Side-Channel Correlation Collision Univariate Attacks.



(b) 2nd-Order Side-Channel Correlation Collision Univariate Attacks.



(c) 3rd-Order Side-Channel Correlation Collision Univariate Attacks.

Figure 5.14: 2nd-Order $\mathsf{GF}((2^4)^2)$ Masked Implementation, Masks On, Shuffling Off, 1st-Order (top), 2nd-Order Side-Channel Correlation Collision Univariate Attacks (middle), 3rd-Order Side-Channel Correlation Collision Univariate Attacks (bottom), 10,000,000 Traces, $f_{\text{clk}} = 4\,\text{MHz}$. The correlation coefficient for the correct key is marked in **black**, while **gray** is used to depict wrong key hypotheses.

6 Defeating the Super-Exponential Security of LRPRFs with Localized EM Attacks

Contents

While masking schemes allow to increase the security of cryptographic implementations to a certain extent, they still come with some major drawbacks: they can be broken by higher-order attacks, they typically have significant performance penalties, and require a good source of randomness to be effective against side-channel attacks. These issues represent a real problem on all those platforms which are resource-constrained, lack a good source of randomness, or can only generate random numbers with low throughput.

A new line of research, called *leakage resilient cryptography*, has been initiated by Stefan Dziembowski and Krzysztof Pietrzak in 2008 [45] to devise cryptographic primitives which can be proven secure even in presence of side-channel information leakage. These new constructions naturally raise interesting questions regarding the practical relevance of these formal models to secure actual physical implementations.

Subsequent works like [117, 74, 76] have tried to answer these questions by analyzing implementations of leakage resilient constructions against concrete side-channel attacks. In particular, Marcel Medwed and François-Xavier Standaert and Antoine Joux [76] showed that LRPRFs have super-exponential security in respect to classical CPA attacks.

In this chapter, we study the impact of localized EM attacks [105, 60] taking advantage of high-resolution EM micro-probes for attacking LRPRF constructions. We use an FPGA case study to highlight that, while the leakage of LRPRFs remains difficult to exploit by actual univariate side-channel attacks, yet the security of the construction can be reduced down to critical levels, when multivariate attacks are used.

Parts of this chapter have been published in the proceedings of the "Third International Conference on Cryptology and Information Security in Latin America, Latincrypt 2014" [40] and in the "Journal of Cryptographic Engineering (JCEN)" in 2014 [10].

Outline

- **Section 6.1** introduces leakage-resilient cryptography with a particular focus on LRPRFs with application to fresh re-keying schemes.
- **Section 6.2** presents the methodology used to evaluate the security of such constructions on FPGAs in face of high-resolution EM attacks.
- **Section 6.3** presents high-resolution EM attack results for the *univariate* case showing that the actual leakage still remains hard to exploit.
- **Section 6.4** presents high-resolution EM attack results for the *multivariate* case showing that the security can be lowered down to critical levels.
- **Section 6.5** provides a summary of the chapter.

6.1 Leakage-Resilient Cryptography

Leakage resilient cryptography exploits the formalism of modern cryptography to realize cryptographic primitives which are provably secure even in presence of leakages. The main idea is to formally model side-channel leakages in order to design primitives which are secure in these models, hence addressing the side-channel issue at design stage.

More concretely, the approach of leakage resilient cryptography is the following: (1) a security model which includes a class of leakage functions is defined. These functions are typically either bounded or noisy to some extent and the model is typically referred to as the *"leakage model"*. (2) A formal definition of security is provided, *e.g.*, in a game-based indistinguishability fashion, to precisely quantify when a primitive is secure. (3) A cryptosystem is designed and proven secure in the previously defined leakage model according to the previously stated definition of security. Dziembowski and Pietrzak's leakage-resilient cryptography [45] is one of the most investigated models for this purpose, and several proposals of Pseudo-Random Generators (PRGs), stream ciphers, PRFs and Pseudo-Random Permutations (PRPs) have been considered in this setting [42, 47, 89, 117, 128, 129].

In practice, leakage-resilient cryptography suffers from three major drawbacks when it comes to implementations: (1) most of the primitives have significant performance overheads, (2) it is not clear how to bridge the gap between the formalism and the practice, *e.g.*, it is not clear how to exactly bound the leakage information in practice, and (3) compatibility with existing cryptographic standards like AES or PRESENT is typically lost.

Yet, they are all based on some kind of re-keying strategies which are reminiscent of Kocher's early patent [67]. Indeed, it might very well be that small variations of the ideas proposed in such theoretical works actually provide significantly security enhancements against a large class of "practical attacks". This naturally suggested an intermediate line of research, where leakage-resilient primitives are used for re-keying schemes and their security is analyzed in front of actual side-channel adversaries, in order to mitigate these overheads and guarantee compatibility with existing cryptographic standards.

In the next sections the following topics are introduced: PRFs, PRGs, a construction of PRFs from PRGs, LRPRFs, and how to use LRPRFs for fresh re-keying schemes in order to ensure backward compatibility with existing cryptographic standards.

6.1.1 Pseudo-Random Functions

PRFs represent one of the most important primitives in symmetric cryptography. Informally speaking, a PRF is a function that is computationally indistinguishable from a random function in the sense that no efficient adversary can distinguish a

PRF from a *truly* random function.

Let a PRF family be denoted as $\mathcal{F}$, where $k \overset{\$}{\leftarrow} \mathcal{U}_\kappa$ is an integer key that selects a random member from $\mathcal{F}$, being a function $\mathbb{F}_k : \{0,1\}^m \rightarrow \{0,1\}^n$, where m, n are fixed input and output sizes. Then, the security of PRFs can be defined in a game-based fashion, where an adversary A is asked to distinguish between the output of a pseudorandom function $\mathbb{F}_k$ and a truly random value.

The adversary A is composed of two algorithms (A_1, A_2) and take its decision by running two formal experiments $\mathbf{Exp}_{\mathsf{A}}^{\text{PRF-IND-0}}$ and $\mathbf{Exp}_{\mathsf{A}}^{\text{PRF-IND-1}}$. The two experiments consist of a *profiling phase*, where the algorithm A_1 adaptively generates the input values for which PRF evaluations are collected, and a *decision phase*, where the algorithm A_1 outputs yet another fresh input to $\mathbb{F}_k$ and the algorithm A_2 is requested to distinguish between the evaluation of the PRF on this last input value and a truly random value. This is formalized in Definition 3 and illustrated in Figure 6.1.

Definition 3 (Pseudorandom function (PRF)) *A function family $\mathcal{F}$ is (q, τ, ε)-*secure *if an adversary $\mathsf{A} = (A_1, A_2)$ that runs in time τ, given an oracle for a function $\mathbb{F}_k$ and allowed to make at most q queries to $\mathbb{F}_k$ has advantage at most ε in distinguishing the output of $\mathbb{F}_k$ from random:*

$$\mathsf{Adv}(\mathbf{Exp}_{\mathsf{A}}^{\textit{PRF-IND}}) := \left| \Pr(\mathbf{Exp}_{\mathsf{A}}^{\textit{PRF-IND-0}} = 1) - \Pr(\mathbf{Exp}_{\mathsf{A}}^{\textit{PRF-IND-1}} = 1) \right| < \varepsilon \tag{6.1}$$

6.1.2 The GGM Construction

Back in 1986, Oded Goldreich, Shafi Goldwasser, and Silvio Micali showed how to construct PRFs from PRGs [55].

Informally speaking, a PRG is an efficient deterministic function that expands a short and *truly* random seed into a polynomially long *pseudorandom* sequence, which cannot be efficiently distinguished from a truly random one. A formal definition is given in Definition 4.

Definition 4 (Pseudorandom Generator (PRG)) *Let an integer $\kappa \in \mathbb{N}$ be given, and take $\varepsilon, \tau > 0$. A function $\mathsf{G} : \{0,1\}^\kappa \rightarrow \{0,1\}^{s(\kappa)}$ is an (s, τ, ε)-*secure PRG *if:*

1. *G is computable in deterministic polynomial time in the input length κ.* [Efficiency]
2. *There exists a polynomial s, called the* stretch, *such that:* [Regular stretch]
    - $s(\kappa) > \kappa$
    - $|\mathsf{G}(x)| = s(\kappa)$ *for all strings $x \in \{0,1\}^\kappa$.*

```
$\mathbf{Exp}_{\mathsf{A}}^{\text{PRF-IND-0}}$ :                          $\mathbf{Exp}_{\mathsf{A}}^{\text{PRF-IND-1}}$ :
    $k \xleftarrow{\$} \{0,1\}^{\kappa}$                                  $k \xleftarrow{\$} \{0,1\}^{\kappa}$                       [Initialization]
    $S, I \leftarrow \emptyset$                                        $S, I \leftarrow \emptyset$
    for $i = 1, 2, \ldots, q$                                  for $i = 1, 2, \ldots, q$                     [Perform $q$ queries]
        $x_i \leftarrow A_1(S, I)$                                  $x_i \leftarrow A_1(S, I)$                     [Generate inputs]
        $S \leftarrow S \cup \{x_i\}$                                  $S \leftarrow S \cup \{x_i\}$                  [Collect inputs in $S$]
        $I \leftarrow I \cup \{(x_i, \mathbb{F}_k(x_i))\}$                      $I \leftarrow I \cup \{(x_i, \mathbb{F}_k(x_i))\}$   [Collect outputs in $I$]
    endfor                                             endfor
    $x_{q+1} \notin S \leftarrow A_1(S, I)$                              $x_{q+1} \notin S \leftarrow A_1(S, I)$            [Generate challenge]
    $\mathbf{z} \leftarrow \mathbb{F}_{\mathbf{k}}(\mathbf{x_{q+1}})$                                  $\mathbf{z} \xleftarrow{\$} \mathcal{U}_{\kappa}$                          **[Generate response]**
    $b \leftarrow A_2(x_{q+1}, z, I)$                                $b \leftarrow A_2(x_{q+1}, z, I)$                [Take a decision]
    return $b$                                         return $b$
```

Figure 6.1: Definition of a PRF using an Indistinguishability Game: Note that the two experiments differ only in the way the response z is generated. This is generated using either a PRF (left) or a random draw from the uniform distribution (right).

3. *The distribution $\{x \xleftarrow{\$} \mathcal{U}_{\kappa} \colon G(x)\}$ is computationally indistinguishable from the uniform distribution $\mathcal{U}_{s(\kappa)}$ on $s(\kappa)$-bit strings to any efficient adversary $\mathsf{A}(x)$ running in time at most τ:* [Pseudorandomness]

$$|\Pr[x \xleftarrow{\$} \mathcal{U}_{\kappa} \colon \mathsf{A}(\mathsf{G}(x)) = 1] - \Pr[x \xleftarrow{\$} \mathcal{U}_{s(\kappa)} \colon \mathsf{A}(x) = 1]| \leq \varepsilon \tag{6.2}$$

In the remainder of this chapter, this PRF construction will be referred to as the Goldreich-Goldwasser-Micali (GGM) construction after the surnames of its authors.

The GGM construction is essentially a tree traversal through several PRG calls as formalized in Definition 5 for the case 2^b with $b \geq 1$. Using $b > 1$ allows to traverse the tree from the root to the leaves in logarithmic time, hence providing a significant speed-up when it comes to consider the performance of GGM-tree implementations.

Note that the original construction of [55] was described for $b = 1$ only.

Definition 5 (GGM Construction) *Let κ be a security parameter and* $\mathsf{G} : \{0,1\}^{\kappa} \rightarrow \{0,1\}^{2^b\kappa}$ *be a* $(2^b\kappa, \tau, \varepsilon)$*-secure* PRG. *Let* $g_j(x)$ *denote the* $(j+1)^{th}$ κ*-bit output block of* $\mathsf{G}(x)$ *for* $j = 0, \ldots, 2^b - 1$*, and fix a positive constant* $m \in \mathbb{N}$. *Let* $x \in \{0,1\}^m$ *be an input message, which can be partitioned into n blocks of b-bit,* i.e., $x = x_1||x_2||...||x_n$*, and* $k \in \{0,1\}^{\kappa}$ *a secret value. Then, a PRF* $\mathbb{F}\colon \{0,1\}^{\kappa} \times \{0,1\}^m \rightarrow \{0,1\}^{\kappa}$ *is constructed as follows, using the efficient* GGM*:*

$$\begin{cases} y_0 & = & k & & \text{[Initialization]} \\ y_i & = & g_{x_i}(y_{i-1}) & \textit{for } i = 1, \ldots, n & \text{[Iteration]} \\ \mathbb{F}_k(x) & := & y_n & & \text{[Output]} \end{cases} \tag{6.3}$$

6.1.3 Leakage-Resilient Pseudo-Random Functions

LRPRFs can be implemented using block-ciphers: essentially it consists of a block-cipher in a mode of operation defined by the traversal of 2^b-ary tree according to Definition 6.

Definition 6 (Block-cipher based GGM**-tree PRF)** *Let κ be a security parameter and* $\mathsf{BC} : \{0,1\}^{\kappa} \times \{0,1\}^{\kappa} \rightarrow \{0,1\}^{\kappa}$ *be a SPN-based block-cipher with b-bit S-boxes. Let* $x \in \{0,1\}^m$ *be an input message, which can be partitioned into n blocks of b-bit,* i.e., $x = x_1||x_2||...||x_n$*, and* $k \in \{0,1\}^{\kappa}$ *a secret value. Then, a PRF* $\mathbb{F}\colon \{0,1\}^{\kappa} \times \{0,1\}^m \rightarrow \{0,1\}^{\kappa}$ *is constructed as follows, using the efficient* GGM*-tree construction:*

$$\begin{cases} y_0 & = & k & & \text{[Initialization]} \\ y_i & = & \mathsf{BC}y_{i-1}(\underbrace{x_i||x_i||...||x_i}_{\kappa/b \textit{ times}}) & \textit{for } i = 1, \ldots, n & \text{[Iteration]} \\ \mathbb{F}_k(x) & := & y_n & & \text{[Output]} \end{cases} \tag{6.4}$$

At CHES 2012, Marcel Medwed and François-Xavier Standaert and Antoine Joux [76] showed that if all S-boxes run perfectly in parallel, *i.e.,* there are no time shifts and the leakages are perfectly aligned in time, they all leak information through the same leakage function, and both measurements and attacks lead to perfect results (all key words are ranked first after side-channel attacks), yet it remains a search complexity of $N_s!$ for finding the correct subkeys ordering within the master key, hence leading to search problem with super-exponential complexity in the number of parallel S-boxes N_s. In fact, any divide-and-conquer DPA trying to exploit the leakage of such a construction will be affected by *key-dependent* algorithmic noise, *i.e.,* the leakage will only provide information about the master key up to a permutation of its words (cf. [76] for a detailed analysis of this claim).

In practice, this means that using AES as a building block of a LRPRF would lead to a search complexity of $16! \approx 2^{44}$ in the best case. Since this complexity is still doable with modern technology, this fact suggests that AES may not be the best block-cipher for integration in a leakage-resilient PRF. For this reason, an alternative lightweight design based on the PRESENT block-cipher is considered in this work. In particular, the "identical leakage" assumption is validated in practice in face of high-resolution EM attacks.

6.1.4 Fresh Re-Keying with PRFs

Fresh re-keying schemes have been firstly analyzed in [2] and subsequently implemented in [75, 10]. A fresh re-keying scheme consists of a re-keying function $\mathsf{g} : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^\kappa$, which takes a secret key $k \in \{0,1\}^\kappa$ and a public random nonce $r \in \{0,1\}^n$ as input to generate a session key $k^* \in \{0,1\}^\kappa$ as output, and a cryptographic function $\mathsf{f}(k^*, \cdot)$ which uses the freshly generated key k^* for a certain number of cryptographic operations. For instance, a fresh key can be computed as $k^\star = g_k(r)$, and then used to generate the ciphertext $c = \mathsf{BC}_{k^\star}(p)$ from a plaintext p using a block-cipher BC. This allows to protect standard cryptographic functions such as AES or PRESENT without breaking backward compatibility with existing standards.

The solution proposed in [75] was to use a modular multiplication for g, which benefits from the feature of being easy to mask and shuffle.

Using a leakage resilient PRF for re-keying (and not directly as a primitive) allows to relax its cryptographic properties (and therefore reduce the implementation's requirements) as only minimal diffusion properties might be sufficient for re-keying functions (cf. Section 2.4.3). In this case, it would be more appropriate to refer to such a construction as a leakage resilient PRF-like construction.

More formally, a re-keying schema using a block-cipher $\mathsf{BC} : \{0,1\}^\kappa \times \{0,1\}^\kappa \rightarrow \{0,1\}^\kappa$ as a LRPRF is defined in Equation (6.5), *i.e.*, the tree traversal process begins from the master key k and proceeds as follows: for any given public nonce $r \in \{0,1\}^{nb}$ and key $k \in \{0,1\}^\kappa$, the block-cipher is iterated n times using the previous block-cipher output as a key and the n^{th} word (of size b) of the nonce replicated to cover the full input size κ of the block as input, *e.g.*, $r_0|||r_0||...||r_0$.

$$\begin{cases} y_0 &= k & & \text{[Initialization]} \\ y_{i+1} &= \mathsf{BC}_{y_i}(\underbrace{r_i||r_i||...||r_i}_{\kappa/b \text{ times}}) & \forall i = 0,\ldots,n-1 & \text{[Iteration]} \\ PRF(k,r) &:= y_n & & \text{[Output]} \\ c &= \mathsf{BC}_{PRF(k,r)}(p) & & \text{[Encrypt]} \end{cases} \tag{6.5}$$

6.2 Evaluation Methodology

In this section, the actual LRPRF design used for the side-channel evaluation is presented together with the used measurement setup and a description of the side-channel attacks which were performed.

6.2.1 FPGA Prototype Design

For the purpose of attacks, we considered a prototype leakage resilient PRF implementation on FPGA based on the PRESENT block-cipher.

It implements the S-box layer using $n = 32$ parallel PRESENT S-boxes and the `Single-Pattern` bit-permutation layer which was proposed in [10] for $b = 4$:

$$P(i) = ((i \bmod b) \cdot (n+1) + (\lfloor i/b \rfloor \bmod b) \cdot n + \lfloor i/b^2 \rfloor \cdot b) \bmod (b \cdot n). \tag{6.6}$$

This permutation connects the first bit of each S-box output to the first bit of a word after the permutation, the second bit of each S-box output to the second bit of a word after the permutation, ... Hence, side-channel attacks using the Hamming distance as described in [10] yields no extra information about the location of the key words (cf. [10] for a detailed analysis of this claim).

For the sake of visualization, the `Single-Pattern` permutation layer for $(b, n) = (4, 16)$ is illustrated in Figure 6.2.

In order to allow for a worst-case analysis of our re-keying function, the architecture provides two operational modes.

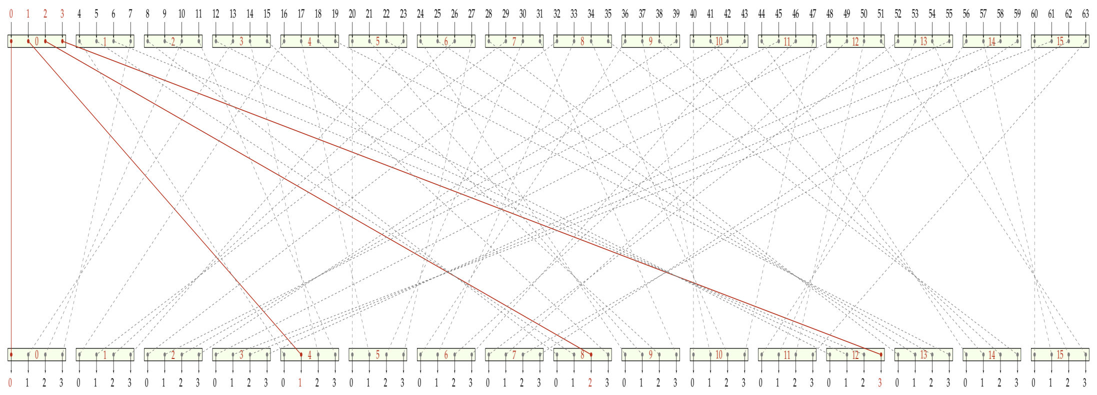


Figure 6.2: `Single-Pattern` permutation layer for $(b, n) = (4, 16)$ keeping the bit-ordering within the input and output nibbles: the four input bits of an input nibble are connected to four different output nibbles, where the first bit of the input nibble is connected to the first bit of an output nibble, the second bit of the input nibble is connected to the second bit of an output nibble, the third bit of the input nibble is connected to the third bit of an output nibble, and the fourth bit of the input nibble is connected to the fourth bit of an output nibble. This pattern is marked with **red** for the first nibble.

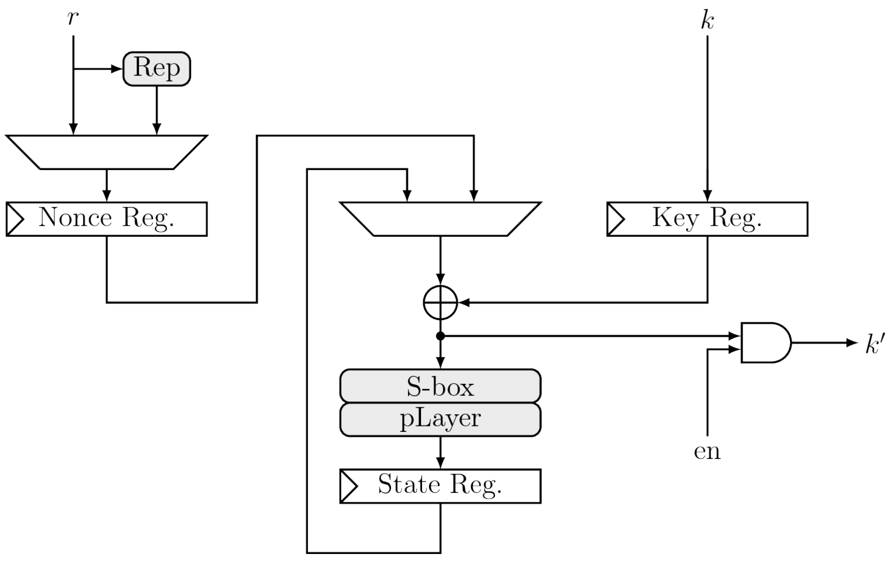


Figure 6.3: Prototype architecture for worst-case EM profiling.

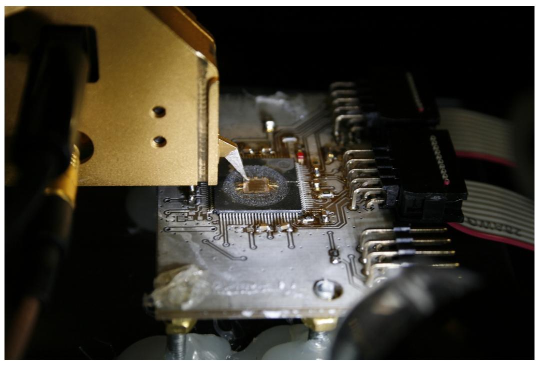
In the first (open) mode, it is possible to change each single word of both the master key k and the nonce r, while keeping all the other words constant. Although this is exactly what is prevented by the construction, *i.e.*, only carefully selected plaintexts should be observable by the adversary, this mode was investigated in order to allow for a profiling phase without the impediment of the key-dependent algorithmic noise. This mode corresponds to the typical block-cipher in a standard side-channel attack scenario.

In the second (fixed) mode, the master key is fixed and each word r_i of the nonce in the i^{th} step is replicated 32 times to cover the length of the nonce register. This corresponds to the actual circumstances that an adversary would face when attacking a leakage-resilient PRF. The prototype architecture is shown in Figure 6.3.

6.2.2 Measurement Setup

Our target device is a Xilinx Spartan-3A (XC3S200A) FPGA manufactured in a 90 nm process technology. The VQ100 package was opened as described in Section 2.2.3 to allow for front-side high-resolution EM field measurements over the surface of the chip die. The setup is shown in Figure 6.4.

We used an inductive near-field EM probe (ICR HH 150-6 from Langer EMV-Technik[1]) with a 150 µm shielded horizontal coil, 6 windings, and 2.5 MHz − 6 GHz frequency span. This probe contains an horizontal magnetic loop which can measure the vertical components of the superposed magnetic field generated by the target chip die.


(a) EM micro-probe and target decapsulated FPGA device.

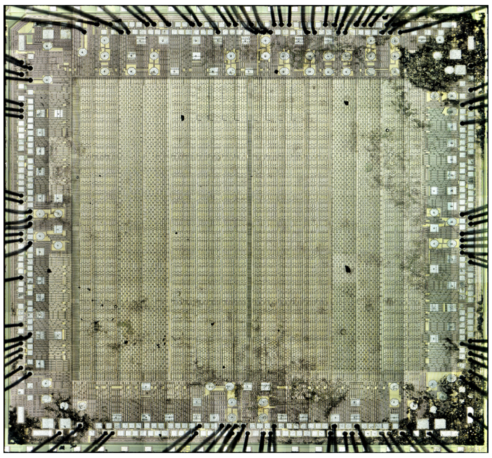

(b) FPGA Die.

Figure 6.4: Measurement Setup used for high-resolution EM analysis of our LRPRF: EM micro-probe on the target device (left) and a photo of the FPGA die obtained using microscope USB camera (right).

[1]`https://www.langer-emv.de`

We connected the probe to an external 30 dB amplifier (PA 303 BNC Set Pre-amplifier from Langer EMV-Technik) and step with a resolution of 100 µm all over the 2700 µm $\times$ 2700 µm die surface of the FPGA.

A LeCroy WavePro 725Zi oscilloscope with 2.5 GHz bandwidth and a sampling rate of 5 GSa/s was used to record the measurements. The FPGA is externally clocked at 20 MHz and synchronized with the PLL of the oscilloscope.

The *unconstrained* placement of our circuit on the floorplan is shown in Figure 6.5b. We performed 27×27 measurements covering the surface area confined by the conjunctions of the bonding wires using a motorized X-Y stepper.

6.3 Univariate Localized EM Attacks

In this section, we describe the worst-case profiling together with the selection of POIs in the EM maps. Also, we present the results of the attacks against the implementation operating in fixed mode and taking advantage of these worst-case profiles.

6.3.1 Worst-case profiling in open mode

In open mode, the adversary is able to independently observe the EM leakage characteristic of each subkey at different locations over the chip surface, without the influence of the key-dependent algorithmic noise (since the untargeted nibbles can be set to random values).

Hence, the attacker can directly profile a leakage model of each subkey, just as in any other parallel implementation. In order to identify the univariate leakage of individual subkeys, we recorded 2^{16} measurements and computed the SNR for each nibble $j \in [0, 31]$, at each location (x, y) and for each time instant t. That is, $\mathrm{SNR}_j(x, y, t) = \frac{\hat{\sigma}(\hat{\mu}_{0\to0}, \hat{\mu}_{0\to1}, \ldots, \hat{\mu}_{F\to F})}{\hat{\mu}(\hat{\sigma}_{0\to0}, \hat{\sigma}_{0\to1}, \ldots, \hat{\sigma}_{F\to F})}$, where $\hat{\mu}_{u\to w}$ and $\hat{\sigma}_{u\to w}$ are the maximum likelihood estimators of the mean value and standard deviation of the leakages at time instant t conditioned on the transition from the value u to the value w of the target S-box.

The 4-bit inputs to the key and nonce registers were carefully chosen from a 16-bit Linear Feedback Shift Register (LFSR), in order to produce all the possible 256 transitions of a nibble in the state register exactly 256 times each. As a result, we obtained 32 SNR maps which are shown Figure 6.6 and Figure 6.7.

It can be observed that the leakage of individual key nibbles are clearly bounded to some confined regions on the chip surface. However, if we consider the leakage of each subkey as occurring simultaneously during an actual attack, then all SNRs overlap significantly, as shown in the upper part of Figure 6.5.

This result can be easily explained by looking at the placement of our design on the floorplan, which is shown in Figure 6.5b. In fact, contrary to [61] where constraints on the placement were set, in our case the logic cells on the floorplan of the FPGA are located only in one large fuzzy region due to an *unconstrained* placement. This region overlaps with the region of high SNR.

Given these preliminary results, the next question is to determine how to select the POIs that will be used in our attacks. Quite naturally, the previous SNRs considered individually are not optimal in this respect, since they are based on the implicit assumption of independent (algorithmic) noise.

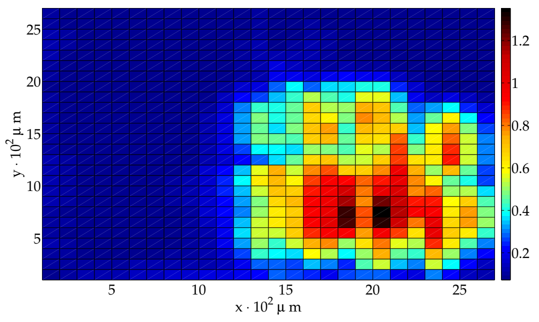
Therefore, we considered two additional criteria in order to better reflect the activity of individual key nibbles considering the presence of key-dependent algorithmic noise, namely:

$$C_2 = \underset{(x,y,t)}{\arg\max} \frac{\mathrm{SNR}_j(x,y,t)}{\sum_{i\neq j}\mathrm{SNR}_i(x,y,t)}, \quad C_3 = \underset{(x,y,t)}{\arg\max} \frac{\mathrm{SNR}_j(x,y,t)}{\max_{i\neq j}\mathrm{SNR}_i(x,y,t)}. \tag{6.7}$$

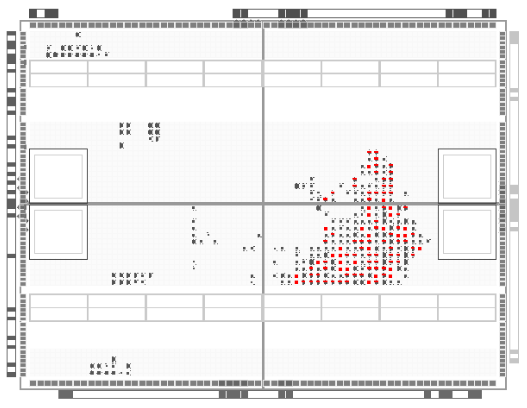
The intuition behind these criteria is that the best POIs should isolate one target S-box from either all the other S-boxes (on average) or from the "closest" S-box.

6.3.2 Attacks exploiting worst-case profiles in fixed mode

For the different selections of POIs presented in the previous subsection (including the basic SNR), we built leakage models and then performed 32 CPA attacks in fixed mode, using a fresh set of measurements. In this context, the data complexity for each attack is bounded to 16. Yet, nothing prevents an adversary to repeatedly measure each of its allowed input queries in order to get rid of physical noise. Hence, we performed attacks exploiting increasing number of traces (from 2^8 to 2^{16}) and first observed that the results were stable from 2^{12} traces on. Next, we looked at the subkey ranks, *i.e.,* the position of the correct subkeys in the 32 vectors of 16 candidates as provided by the attacks. For illustration, we list the ones obtained for the worst criteria (SNR) and the best one, *i.e.,* C_2 or C_3 depending on the S-boxes:



(a) Superimposition of the SNR for all the S-boxes over the 27 × 27 chip surface. The SNR is located in the same region of the placed circuit (lower right corner).



(b) Placement of our implementation on the FPGA floorplan obtained *without* enforcing placement constraints in the synthesis: Flip-Flops of the working register are marked in **red**.

Figure 6.5: Superposition of the SNR for all the S-boxes over the 27 × 27 chip surface (top). Placement of our implementation on the FPGA floorplan (bottom).

Table 6.1: Ranking of all 32 key-nibbles obtained for the worst criteria (SNR) and the best one, *i.e.*, C_2 or C_3 depending on the S-boxes.

SNR	1	5	14	7	6	8	3	1	2	1	1	14	14	1	7	1
	6	9	6	15	6	1	1	3	6	16	7	14	8	2	11	1
Best	1	1	14	2	3	4	1	1	2	1	1	7	14	1	7	1
	6	6	6	12	1	1	1	3	3	6	2	12	8	1	7	1

One can directly observe that for a number of S-boxes (namely, 9 for the worst and 13 for the best cases), the correct subkey is ranked first - hence suggesting that the high-resolution EM profiling indeed allows for a separation of the S-box leakages, thus leading to improved attacks.

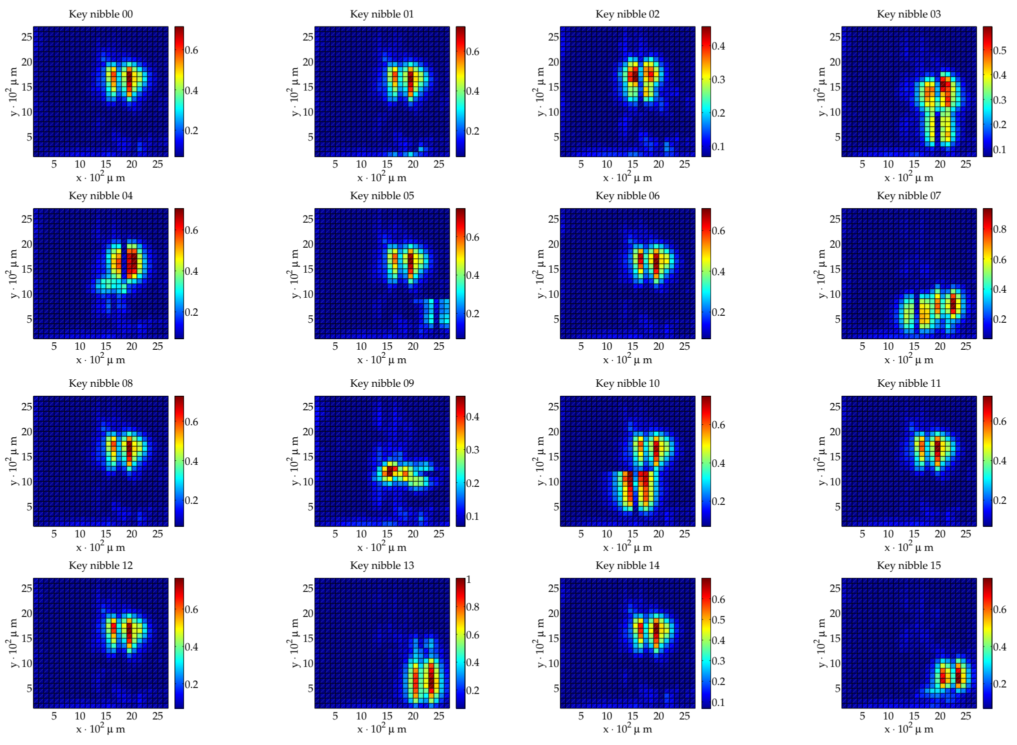


Figure 6.6: SNR Maps for the Key Nibbles 0-15: each key nibble has a different (almost unique) SNR pattern.

Yet, looking at the CPA results more precisely, we also observed that firstly ranked subkeys were usually slightly better correlated than other candidates. By contrast for badly ranked subkeys, some of them showed very poor correlation results. The main consequence of this observation is that enumerating the master key remains a computationally intensive task, even in the context where worst-case profiling is possible. To illustrate this claim, first observe that an underestimated time complexity for the enumeration can be obtained by computing the product of the subkey ranks. From the two previous lists, we obtain 2^{64} and 2^{46}, respectively. Improving this lower bound can be done by merging the lists, *e.g.*, the 16 subkey ranks for 8-bit values corresponding to the same two attacks (aggregated) are given by:

Table 6.2: Ranking of all 16 key-bytes obtained for the worst criteria (SNR) and the best one, *i.e.*, C_2 or C_3 depending on the S-boxes.

SNR	9	202	59	9	7	68	78	26	90	159	6	11	142	112	80	78
Best	1	76	19	1	7	27	78	26	50	107	1	11	35	50	43	36

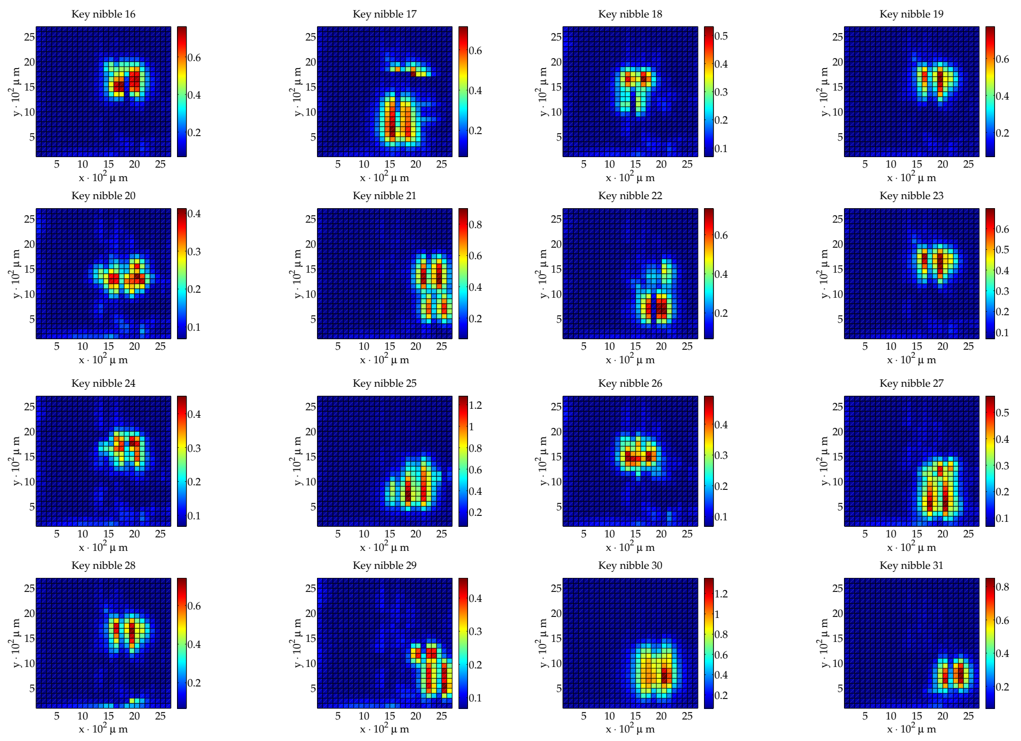


Figure 6.7: SNR Maps for the Key Nibbles 16-31: each key nibble has a different (almost unique) SNR pattern.

leading to refined bounds of 2^{86} and 2^{66}, respectively.

Intuitively, the better bounds derive from the fact that when merging dimensions (as an optimal key enumeration algorithm does [122]), the time complexity significantly increases every time both subkeys are not highly ranked.

Using the rank estimation algorithm in [123], we finally obtained tight bounds for the master key rank as $[2^{115} - 2^{118}]$ and $[2^{99} - 2^{102}]$. Quite naturally, one could further consider that the knowledge of which subkeys are "easy to recover" is an additional outcome of the worst-case profiling[2]. In this conservative scenario, the adversary could reduce the dimension of the enumeration problem (down to 23 and 19, respectively), but our experiments still lead to security bounds of $[2^{89} - 2^{90}]$ and $[2^{69} - 2^{70}]$.

[2]This is realistic as this information mainly depends on the placement of the S-boxes in the implementation. By contrast, the information of the correct subkey ranking depends on the key-dependent algorithmic noise and cannot be considered as constant for all attacks.

6.4 Multivariate Localized EM Attacks

Side-channel attacks generally rely on the hypothesis that the leakage information is located in some certain points of the measurements which are initially unknown to the adversary (POIs). In case of high-resolution EM measurements, such points can be located either in time or in space, *i.e.,* at a certain position over the die surface. There are basically two approaches to select POIs: either some heuristic method is used, like SNRs, or some linear combination of the leakages is used, *e.g.,* based on either Principal Component Analysis (PCA) or the Fisher's linear discriminant [4, 28].

For the purpose of attacks, we performed Template Attacks (TAs) using the heuristics of Equation (6.7) and then used PCA to transform the concatenated measurements into the principal subspace and perform PCA-based Template Attacks (PCATAs) as described in [115].

Following [28], we built templates using PCA pre-processing as follows:

1. We computed the sample between groups matrix **B**:

   $$\mathbf{B} = n_p \sum_{\tau \in \mathcal{S}} (\boldsymbol{\mu}_\tau^x - \boldsymbol{\mu}_x)^T (\boldsymbol{\mu}_\tau^x - \boldsymbol{\mu}_x),$$

   where n_p is the number of leakage traces per template τ in the set of all templates $\mathcal{S}$, $\boldsymbol{\mu}_x = \frac{1}{S}\sum_{\tau \in \mathcal{S}} \boldsymbol{\mu}_\tau^x$, $\boldsymbol{\mu}_\tau^x = \frac{1}{n_p}\sum_{n=1}^{n_p} \boldsymbol{x}_n^\tau$ and $\boldsymbol{x}_n^\tau$ is the n^{th} leakage trace for the template $\tau \in \mathcal{S}$.

2. We obtained the Singular Value Decomposition (SVD) $\mathbf{B} = \mathbf{U}\mathbf{D}\mathbf{U}^T$, where each column of $\mathbf{U}$ is an eigenvector $\mathbf{u}_j$ of $\mathbf{B}$ and $\mathbf{D}$ contains the corresponding eigenvalues on its diagonal.

3. The PCA projection matrix was obtained as $\mathbf{U}_c = [\mathbf{u}_1, \ldots, \mathbf{u}_c]$, where only the first c components of $\mathbf{U}$ are retained using the elbow rule [28].

4. Finally, the projected leakage matrix in the principal components $\mathbf{X}_{\tau,c} = \mathbf{X}_\tau \mathbf{U}_c$ and templates were built.

We used the rank estimation algorithm [123] to estimate tight security bounds after TAs and PCATAs for our considered prototype FPGA implementation. The rankings for the correct subkeys obtained from these attacks are reported in Table 6.3.

Using TAs and PCATAs more subkeys were ranked first with an overall improvement in the ranking for all subkeys, hence leading to refined security bounds of $[2^{107} - 2^{110}]$ and $[2^{83} - 2^{86}]$, respectively.

Table 6.3: Ranking of all 32 key-nibbles using TA and PCATA.

TA	1	3	14	7	6	7	3	1	1	1	1	12	14	1	7	1
	6	9	5	14	6	1	1	2	6	15	7	12	8	1	10	1
PCATA	1	1	14	1	1	4	1	1	1	1	1	7	14	1	6	1
	4	6	5	12	1	1	1	1	3	5	1	10	7	1	7	1

Assuming the conservative scenario, where the knowledge of which subkeys are "easy to recover" is assumed, the adversary could reduce the dimension of enumeration further down to $[2^{83} - 2^{84}]$ and $[2^{55} - 2^{56}]$, respectively. These security bounds can not to be considered satisfying anymore in light of modern computing capabilities.

6.5 Summary

Fresh re-keying using leakage-resilient PRFs provides an interesting approach to secure embedded systems against global side-channel attacks. One of the main advantages is that the side-channel security of the LRPRFs does not depend on any sources of randomness, but it relies on sound construction principles.

The security of LRPRFs was firstly evaluated in [76]. In their work, the authors showed that the security of AES-128 based leakage-resilient PRFs is limited to 2^{44}, while Rijndael-192 and Rijndael-256 based constructions have a security of 2^{79} and 2^{117}, respectively. Since these latter have prohibitive area and power requirements for resource-constrained embedded devices, we considered a more lightweight block-cipher, which uses a 4-bit PRESENT S-box and a PRESENT-like permutation layer with a security of $32! \approx 2^{117}$.

Using the key rank estimation algorithm, we showed that the remaining security after univariate high-resolution EM attacks on an FPGA implementation is within $[2^{99} - 2^{102}]$, where the POI were selected using different SNR-based criteria. Furthermore we showed that, when the worst-case (though, realistic) scenario was considered, where the previous knowledge about the subkeys which are easier to recover is assumed, then the security bounds were reduced to $[2^{69} - 2^{70}]$.

Finally, when multivariate attacks based on PCA pre-processed measurements were considered, the obtained security bounds were further reduced down to critical levels, namely $[2^{83} - 2^{86}]$ and $[2^{55} - 2^{56}]$, which represent the standard and worst case for the considered FPGA implementation, respectively.

7 Spatial Randomized Leakage-Resilient PRFs

Contents

Leakage-Resilient PRFs represent a very appealing alternative to more common countermeasures like masking or shuffling schemes, due to their super-exponential security against global DPA attacks and their unique feature of not relying on additional randomness to achieve side-channel resistance.

Although the understanding and security evaluation of this type of constructions is still far from the ones of common side-channel protections, such as masking and shuffling, the previous chapter already showed that their security can be lowered down to critical levels when subjected to advanced high-resolution attacks using EM micro-probes over the die surface of a depackaged FPGA.

In this chapter, we propose a countermeasure to thwart high-resolution EM attacks based on spatial randomization and permutation networks. The idea of spatial randomization has been initially proposed in [78] to exploit the dynamic reconfiguration capabilities of FPGAs.

In the context of ASIC design, spatial randomization has been applied to DES in [92]. Their approach is based on simple cyclic shifts and allows for 2^3 permutations only. In contrast, our approach is based on permutation networks and allows for a much larger number of permutations so to *significantly* increase the measurement effort which is necessary to mount successful side-channel attacks.

Outline

- **Section 7.1** introduces the necessary background information on permutation networks by evaluating various alternatives for spatial randomization, namely: Butterfly, Omega, Flip, and Beneš networks.
- **Section 7.2** proposes a generic countermeasure based on spatial randomization and discusses the performance of different implementations based on permutation networks when applied to leakage-resilient PRFs.
- **Section 7.3** describes the hardware design of LRPRFs with spatial randomization and discusses implementation results.
- **Section 7.4** provides a summary of the chapter.

7.1 Evaluation of Permutation Networks

Let $\mathcal{U}_m = \{0,1\}^m$ be the set of strings of length m. A (n,m)-random permutation is a permutation $\sigma_r \in S_n$ associated to a uniformly drawn random index value r:

$$r \overset{\$}{\leftarrow} \mathcal{U}_m : \sigma_r \in S_n,$$

where S_n is the symmetric group of all permutations. Permutation networks can be used in order to approximate (n,m)-random permutations [48].

Permutation networks are based on the observation that transpositions can generate the symmetric group S_n of all permutations. Indeed, the building block of permutation networks is the *switch* element which conditionally swaps two inputs a and b depending on a common control signal c, as depicted in Figure 7.1.

In the context of cryptography, permutation networks have been used to scramble on-chip data [23] and for implementing arithmetic operations in permutation groups in hardware [113].

Permutation networks are formally defined in Definition 7.

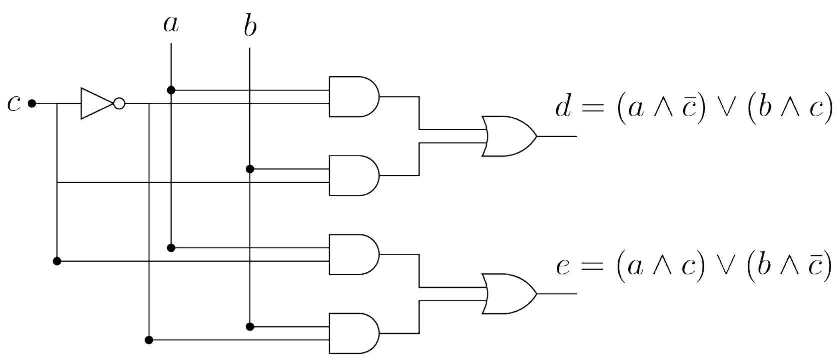

a
b
c
$d = (a \wedge \bar{c}) \vee (b \wedge c)$
$e = (a \wedge c) \vee (b \wedge \bar{c})$

Figure 7.1: Two multiplexers controlled by a single control line c realizing a switch element.

Definition 7 ((n, m, d, s)-Permutation Networks) *A (n, m, d, s)-permutation network is a circuit which permutes n inputs using s switches organized into d layers controlled by a random input value $r \overset{\$}{\leftarrow} \{0, 1\}^m$ which defines a random configuration of the switches.*

Permutation networks mainly differ in the number of stages d and in the way inputs are connected to switches in each stage. These topological parameters ultimately define the distribution of the generated permutations.

For our intended application, we consider only permutation networks with $n = 2^k$ inputs for $k \in \mathbb{N}$ and $n/2$ switches per stage ($s = nd/2$ switches in total). The goal is to implement (n, m)-random permutations with the following properties:

Property 1. The number of permutations $S = 2^m$ should be large, *e.g.*, $S \geq 2^{80}$.

Property 2. The distribution of permutations should be uniform:

$$\Pr[\sigma_r] = 1/S$$

Property 3. The distribution of permuted elements should be uniform:

$$\Pr[\sigma_r(i) = j] = 1/n$$

Property 4. The implementation should be efficient, *e.g.*, the circuit should have logarithmic depth.

Property 1. and *Property 2.* together ensure that the effort of side-channel attacks is significantly increased, while *Property 3.* ensures that every permuted element is not occurring more than others so to ensure the same (theoretical) resistance against high-resolution EM field attacks. Finally, *Property 4.* allows to implement the circuit in combinational logic so to avoid intermediate permutation stages which typically lead to undesired leakages (Hamming distances).

In the following Butterfly, Omega, Flip, and Beneš networks are introduced. A summary of their parameters and costs is provided in Table 7.1 and Table 7.2.

7.1.1 Butterfly Networks

Definition 8 (Butterfly Networks) *Let $n = 2^k, k \in \mathbb{N}$. A butterfly network is a $(n, \frac{n \log_2 n}{2}, \log_2 n, \frac{n \log_2 n}{2})$-permutation network where the switches connections in each stage $1 \leq i \leq d$ are given by $S_i = \{(j, j + 2^{k-i} \mod 2^{k-i+1} + 2^{k-i+1}\lfloor j/2^{k-i+1} \rfloor)\}$ for $j = 0, \dots, n-1$. The inverse of a butterfly network is obtained by reversing the order of the stages.*

Butterfly networks have depth $\log_2 n$ and generate $S = n^{n/2}$ permutations with $\Pr[\sigma] = 1/S$ and $\Pr[\sigma(i) = j] = 2^{-k}$, therefore satisfying *Property 4.*, *Property 1.*, *Property 2.*, and *Property 3.*, respectively. For the sake of an example, a butterfly network with $n = 8$ inputs is illustrated in Figure 7.2.

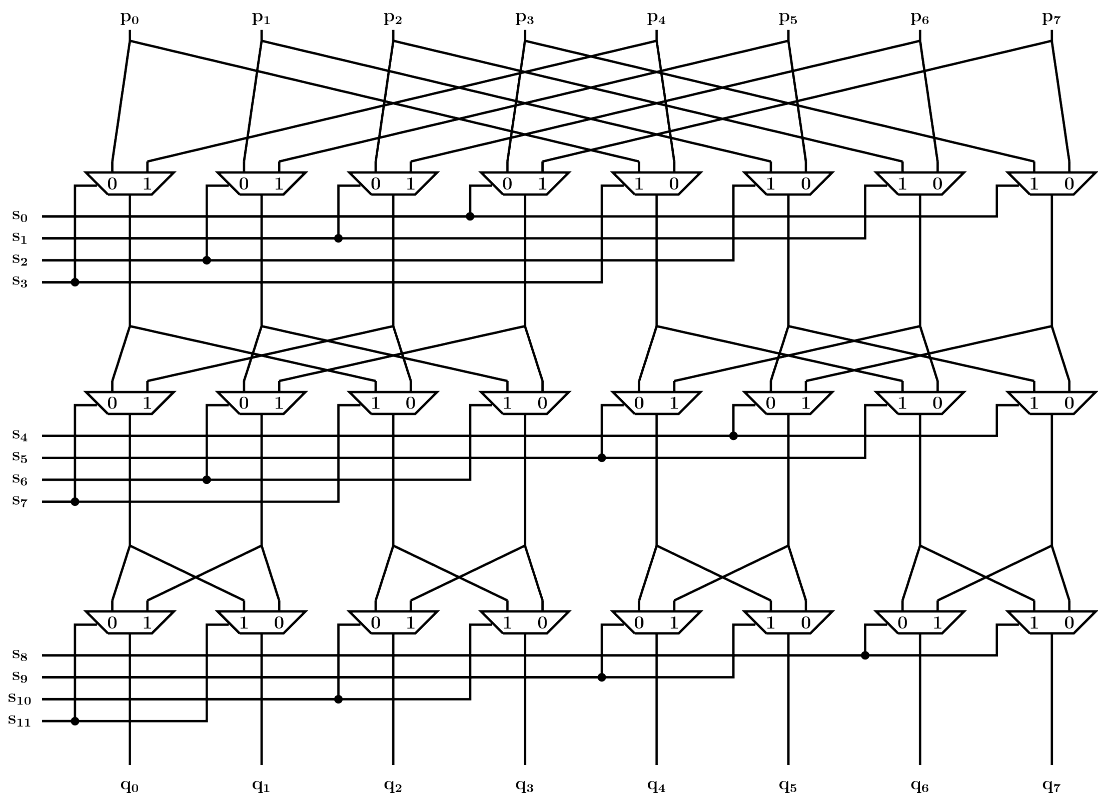
The set of permutations generated by butterfly networks does not form a group under function composition, but it contains the Sylow-2 subgroup $\mathcal{H}_k$ of S_n ($|\mathcal{H}_k| = 2^{n-1}$). The subgroup $\mathcal{H}_k$ can be obtained from butterfly networks by a careful choice of the control bits. Unfortunately, this Group Theoretic (GT) construction [23] leads to a $(n, n-1, \log_2 n, \frac{n \log_2 n}{2})$-permutation network which does not fulfill *Property 1.* for our intended application ($S = 2^{31}$).

Butterfly networks typically have two drawbacks: first, the inputs of each stage are connected to switches in a different way, therefore making it not possible in hardware to fold different stages into a single one. Secondly, butterfly networks are not able to generate all the permutations of S_n, therefore being not suitable for certain applications.

7.1.2 Omega and Flip Networks

A shuffling stage interleaves the two halves of the input bits, while an unshuffling stage performs the inverse operation, as illustrated in Figure 7.3 by the way of an example.

Definition 9 (Omega and Flip Networks) *Let $n = 2^k, k \in \mathbb{N}$. An Omega network is a $(n, \frac{n \log_2 n}{2}, \log_2 n, \frac{n \log_2 n}{2})$-permutation network obtained by iterating $\log_2 n$ times over an Omega permutation stage. An Omega permutation stage is obtained by concatenating*

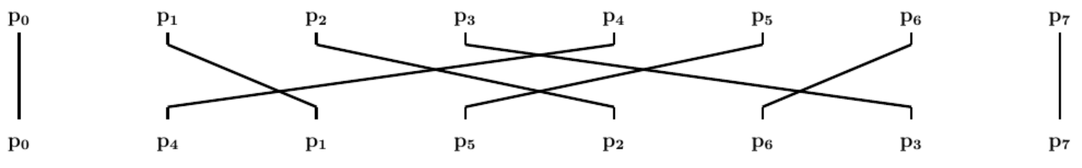


p_0 p_1 p_2 p_3 p_4 p_5 p_6 p_7
0 1
1 0
s_0 s_1 s_2 s_3
s_4 s_5 s_6 s_7
s_8 s_9 s_{10} s_{11}
q_0 q_1 q_2 q_3 q_4 q_5 q_6 q_7

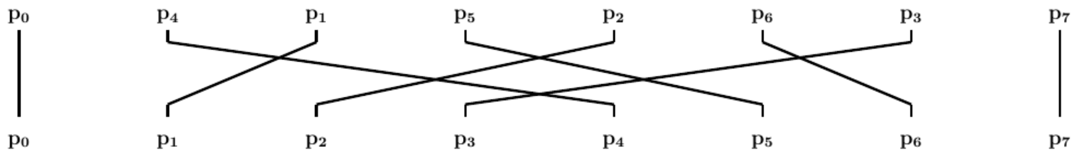
Figure 7.2: Butterfly network with $n = 8$ inputs: this network has $d = 3$ stages and is capable of generating $S = 4096$ permutations.

a shuffling stage with a butterfly stage. The Flip network is a $(n, \frac{n \log_2 n}{2}, \log_2 n, \frac{n \log_2 n}{2})$*-permutation network obtained by iterating* $\log_2 n$ *times over a Flip permutation stage. A Flip permutation stage is obtained by concatenating an unshuffling stage with a butterfly stage. The Flip network is the inverse of the Omega network.*

Omega networks are equivalent to butterfly networks with the advantage that inputs are always connected to switches in the same way across multiple stages. This typically allows for more area-efficient hardware implementations obtained by folding permutation stages into one stage and iterating over it multiple times. However, for our intended application, there is no substantial difference (disregarded from possible synthesis optimizations), as we want to implement permutations networks in combinational logic (fully unrolled). An Omega permutation stage is illustrated in Figure 7.4, while a Flip permutation stage is illustrated in Figure 7.5.



(a) Shuffling Stage.



(b) Unshuffling Stage.

Figure 7.3: Shuffling and Unshuffling Permutation Stages for $n = 8$.

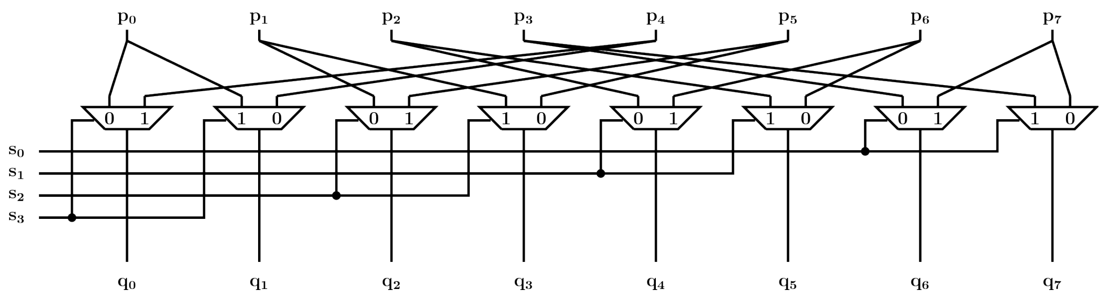


Figure 7.4: Omega Permutation Stage with $n = 8$ inputs: concatenating $d = 3$ Omega permutation stages, an Omega network capable of generating $S = 4096$ permutations is obtained.

7.1.3 Beneš Networks

Definition 10 (Beneš Networks) *Let $n = 2^k, k \in \mathbb{N}$. A Beneš network is a $(n, n\log_2 n - n/2, 2\log_2 n - 1, n\log_2 n - n/2)$-permutation network obtained by concatenating a butterfly network back-to-back to its inverse. Analogously, the concatenation of an Omega and a Flip network is isomorphic to a Beneš network.*

Beneš networks are advantageous over butterfly networks because they can generate $n^n/2^{n/2} > n!$ permutations. As a drawback, the number of switches is almost doubled as well as the number of stages and, most importantly, the distribution of permutations generated from Beneš networks is typically not uniform. Therefore they are not suitable for our intended application, since this feature is violating *Property 2.*.

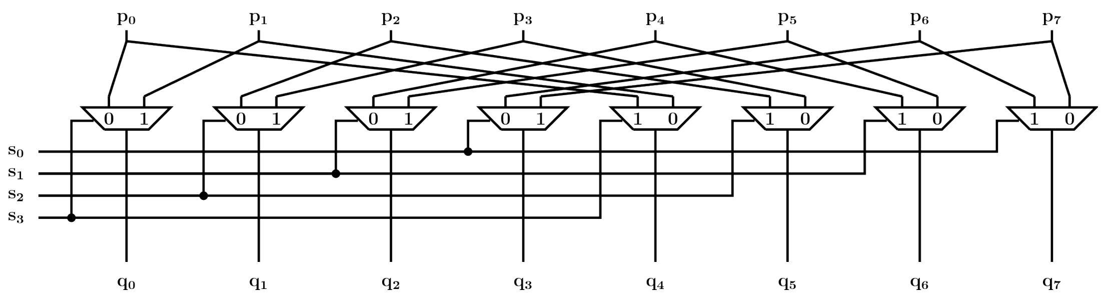


Figure 7.5: Flip Permutation Stage with $n = 8$ inputs: concatenating $d = 3$ Flip permutation stages, a Flip network capable of generating $S = 4096$ permutations is obtained.

7.1.4 Comparison Overview

Table 7.1 and Table 7.2 provide a comparison of the parameters of different permutation networks.

Table 7.1: Permutation Networks: Generic Parameters Comparison for n Inputs.

Permutation Network	# Random bits	# Layers	# Perms
Butterfly (Definition 8)	$\frac{n \log_2(n)}{2}$	$\log_2(n)$	$n^{n/2}$
Butterfly GT (Section 7.1.1)	$n-1$	$\log_2(n)$	2^{n-1}
Omega/Flip (Definition 9)	$\frac{n \log_2(n)}{2}$	$\log_2(n)$	$n^{n/2}$
Beneš (Definition 10)	$n \log_2(n) - n/2$	$2\log_2(n) - 1$	$\frac{n^n}{2^{n/2}}$

Table 7.2: Permutation Networks: Comparison of Concrete Parameters Instantiation for $n = 32$.

Permutation Network	# Random bits	# Layers	# Perms
Butterfly (Definition 8)	80	5	2^{80}
Butterfly GT (Section 7.1.1)	31	5	2^{31}
Omega/Flip (Definition 9)	80	5	2^{80}
Beneš (Definition 10)	144	9	2^{144}

To sum up, Butterfly, Omega and Flip networks satisfy all the properties for $n = 32$ and therefore are suitable for our intended application to spatially randomize the execution order of S-boxes.

7.2 Spatial Randomized Leakage-Resilient PRFs

Spatial randomization is used to physically permute the usage of the logic and wires implementing a cryptographic circuit in order to thwart high-resolution EM field attacks. In principle, one would aim at permuting data before and after every block-cipher operation (key addition, S-box, permutation layer).

However, the task becomes quite challenging if the operations involved operate on different input sizes, as in our considered leakage-resilient PRF (namely, the key addition and the S-box operate nibble-wise, while the permutation layer operates bit-wise). In fact, in order to restore a consistent (nibble-wise) permuted state after the bit-permutation layer, it is necessary to know the (nibble-wise) permutations occurring before entering the bit-permutation layer in order to consistently proceed with next (nibble-wise permuted) key addition after it.

In this situation, two possible alternative approaches exist: either (nibble-wise) permutations are tracked in a way that it is always possible to restore the original state consistently or the original state is simply restored back before the bit-permutation layer and re-permuted nibble-wise right after it.

The first approach allows to persistently keep the state permuted across all the block-cipher operations, but it requires additional and quite complex circuitry for keeping track of (nibble-wise) permutations and it must to be designed ad-hoc depending on the specific properties of the mixing layer and generated permutations. The effort required to keep track of permutations in terms of control circuitry depends on the type and the number of permutations which are allowed.

The second approach requires no additional control circuitry and works independently from the type and number of (nibble-wise) permutations and chosen bit-permutation layer, but it requires to permute the data back to their original (not permuted) state before going through the bit-permutation layer and permute them again right after it.

7.2.1 The Simple Way: Two Stage Rotations

A simple and ad-hoc solution to perform random permutations would be to take advantage of the structure of the bit-permutation layer. The permutation layer of Equation (6.6) groups the n input words of size b into b groups, each one containing n/b words. In other words, it groups the 32 nibbles into 4 groups of 8 nibbles, for $n = 32$ and $b = 4$ as in our case. Therefore, assuming the first stage (left) rotates the state according to a random value $q \overset{\$}{\leftarrow} \{0,1\}^3$ and the second stage (left) rotates the state according to a random value $t \overset{\$}{\leftarrow} \{0,1\}^2$ (say, considering the case where the nibbles of all 4 groups gets rotated by the same value), the permutation layer can be changed as follows in order to produce a (nibble-wise) permuted state after the original linear mixing layer:

$$P(i) = b\left(\underbrace{b\left[\left(\left\lfloor\frac{i}{b^2}\right\rfloor + q\right) \mod \frac{n}{b}\right]}_{\text{Global index}} + \underbrace{\left[\left(\left\lfloor\frac{i}{b}\right\rfloor + t\right) \mod b\right]}_{\text{Local index}}\right) + i \mod b$$

However, the number of permutation $S = 2^5$ is quite limited and therefore not suitable for our application, due to violation of *Property 1.* (allowing independent permutation in the second stage would lead to 2^{11} with additional control logic). This approach actually turns out to be very similar to the approach used in [92] with the same limitations.

7.3 Hardware Design

In order to generate a larger number of permutations, a permutation network can be used. Unfortunately, due to the large number of possible permutations, no efficient way of tracking generated permutations in order to correctly restore a consistent (nibble-wise) permuted state after the bit-permutation layer is known yet.

In this case, the (nibble-wise) permuted state must be restored to its original (not permuted) state before going though the bit permutation layer and subsequently (nibble-wise) permuted again. Figure 7.6 and Figure 7.7 show two possible architectures to implement spatial-randomized leakage-resilient PRFs using permutation networks. The main difference between the two alternative architectures is that the simplified architecture (see Figure 7.7) basically only randomizes the register usage, while the other architecture (see Figure 7.6) also protects the key addition and the S-box operations. Considering the architecture depicted in Figure 7.6, the n nibbles of the state are permuted in every round execution using three random permutations, namely σ_q, σ_q^{-1} and σ_t. More precisely, the key nibbles are permuted before the key addition by the random permutation σ_q, while no permutation takes place on the initial state, as it contains replicated words initially. Then, the state gets permuted back after the S-box operation, goes through the bit-permutation layer and gets nibble-wise permuted again by σ_t. In the first round, $q \overset{\$}{\leftarrow} \{0,1\}^m$ and $t \overset{\$}{\leftarrow} \{0,1\}^m$ are two uniformly chosen random values. From the second round on, q takes the value of t in

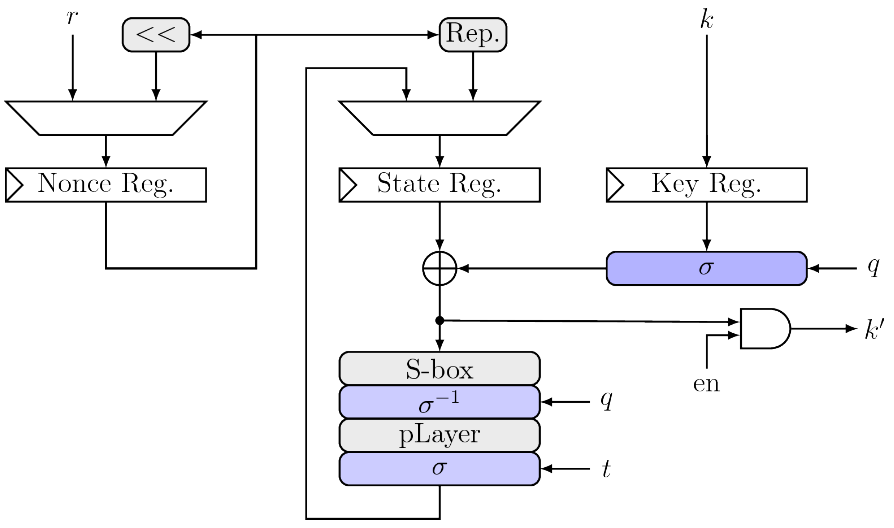


Figure 7.6: Spatial-Randomized Hardware Architecture using Permutation Networks: both the data and key paths are spatially randomized.

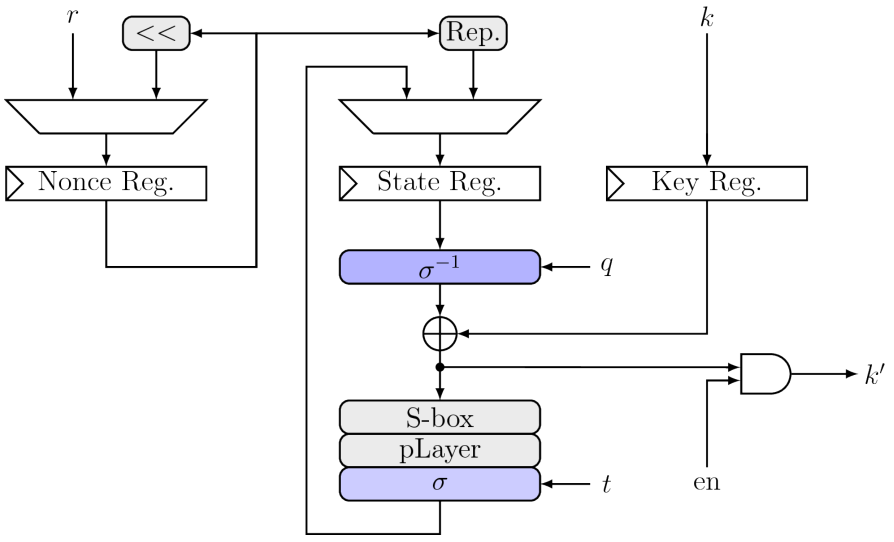


Figure 7.7: Simplified Spatial-Randomized Hardware Architecture using Permutation Networks: only the data path is spatially randomized.

the previous round, while t gets re-freshed by yet another random draw. This means that our countermeasure requires $m(N_rN_s+1)$ additional bits of randomness, where N_r is the number of rounds in the block-cipher and N_s is the number of block-cipher invocations in the PRF construction. The simplified architecture works similarly, but it requires slightly less random bits, namely mN_rN_s.

7.3.1 Implementation Results

The designs were synthesized with *Synopsys DesignCompiler* version *G-2012.06-SP3* using standard cell libraries in 0.18μ CMOS process (csm18os120) and with *Xilinx ISE 14.2* targeting a Xilinx Spartan-3A (XC3S200A).

Table 7.3: Synthesis Results in ASIC and Xilinx Spartan-3A FPGA. The following naming conventions are adopted: the LRPRF implementation without randomization is denoted simply as LRPRF, the LRPRF using two-stage rotations is denoted as LRPRF+2ROT, the LRPRF using a butterfly network for randomization is denoted as LRPRF+BFLY, the LRPRF using an Omega network is denoted as LRPRF+OMEGA, and the LRPRF using the group-theoretic butterfly network is denotes as LRPRF+BFLY-GT.

	FPGA				ASIC	
	Slices	**FFs**	**LUTs**	**f_c (MHz)**	**Area [kGE]**	**f_c (MHz)**
LRPRF	496	397	966	156	4.47	272
LRPRF+2ROT	1645	488	3174	48	10.66	141
LRPRF+BFLY	1392	490	2724	49	7.76	190
LRPRF+OMEGA	1431	490	2806	48	7.76	190
LRPRF+BFLY-GT	1410	435	2748	50	8.47	130

Synthesis results show that the spatial-randomization almost doubles the area requirements in ASIC (while the number of slices and Look-Up Tables (LUTs) is almost triplicated on FPGA), while the maximum clock frequency results affected by almost 1/3 in ASIC and by 2/3 on FPGA.

Interestingly, the requirements of the two-stage rotation based approach are larger than the more generic approach based on permutation networks. This can be explained by the presence of a more complex mixing layer and complex permutation tracking throughout the design in case of the two-stage rotation approach.

Moreover, synthesis results show that LRPRF with the group-theoretic butterfly approach occupies slightly more LUTs than the presumably more complex LRPRF with butterfly one. This is caused by the fact, that LRPRF with the group-theoretic butterfly requires (although fewer) control signals with higher fan-out. This way, extra inverters and buffers are included in the control logic by the synthesis tools.

7.4 Summary

The previous chapter showed that the security of LRPRFs can be lowered down to critical levels when attacked using multivariate attacks employing high-resolution EM micro-probes over the die surface of a depackaged FPGA chip.

In this chapter, we proposed a countermeasure to thwart such high-resolution EM attacks based on spatial randomization and permutation networks to shuffle the circuitry's usage within a PRESENT-like block-cipher. Our countermeasure almost doubles the area requirements of our LRPRF and belongs to the realm of classical side-channel countermeasures which rely on additional randomness bits.

Finally, note that while we applied the spatial randomization countermeasure to protect implementations of leakage resilient PRFs against high-resolution EM attacks, the same countermeasure can also be used to protect parallel implementations of the PRESENT block-cipher or of any other similar SPN-based block-cipher.

8 Conclusion

Nowadays, the security of cryptographic implementations on constrained devices such as RFID tags and smartcards is threatened by side-channel and fault attacks. Two of the most common and effective ways to counteract such attacks are masking and fresh re-keying schemes.

While masking schemes aim at making the side-channel information look random to the adversary, fresh re-keying schemes aim at bounding the side-channel information under a certain threshold which prevents mounting successful attacks.

This thesis aimed at improving state-of-the-art masking and fresh re-keying schemes by providing advancements in the field of side-channel attacks (on the side of adversaries) and side-channel countermeasures (on the side of designers).

Section 8.1 briefly wraps up the major contributions of this work, while Section 8.2 provides some idea sketches for future work.

8.1 Summary

This thesis contains contributions both in the field of secure implementations and side-channel analysis as illustrated in Table 8.1.

Chapter 3 introduced polynomial masking, provided new principles for the selection of implementation-efficient addition chains, described the hardware design of a *higher-order* polynomially masked AES S-box based on our improved addition chains, and finally proposed a shuffling countermeasure which can be easily embedded into polynomial masking schemes at almost no costs.

Chapter 4 showed how tower field arithmetic can be used in combination with polynomial masking to drastically reduce the area size of a polynomially masked AES S-box.

Chapter 5 provided side-channel results using EM measurements of the previously proposed implementations, showing that univariate side-channel security against 10 million traces can be achieved either by a 1st-order polynomially masked implementation using shuffling or by a 2nd-order implementation.

Chapter 6 showed how high-resolution EM measurements can be used to defeat the super-exponential security of leakage-resilient PRFs when using multivariate template-based attacks.

Chapter 7 proposed a shuffling countermeasure to spatially randomize leakage-resilient PRFs using permutation networks as a countermeasure against high-resolution EM attacks.

Table 8.1: Overview on the Contributions of this Thesis.

	Countermeasures	**Side-Channel Attacks**
Chapter 3	Higher-order Polynomially Masked AES S-box with Shuffling	
Chapter 4	Tower-field Polynomially Masked AES S-box	
Chapter 5		Higher-Order EM Analysis of Polynomial Masking with Shuffling
Chapter 6		Localized EM Analysis of a LR-PRF
Chapter 7	Spatial Randomization with Permutation Networks	

8.2 Further Directions

This section provides some idea sketches which can serve as a basis for further research.

Concerning PMSs, the optimality of the methods proposed in Chapter 3 and Chapter 4 could be investigated. Furthermore, the side-channel analysis presented in Chapter 5 could be extended to consider multivariate attacks as well pre-processing functions and windowing methods. Yet another idea would be to investigate the practicability of high-resolution EM attacks in the context of masking schemes.

Concerning LRPRFs, the efficiency of alternative methods for the selection of POIs on the chip's surface, such as Fisher's linear discriminant analysis, could be evaluated. Furthermore, it might be interesting to evaluate the efficiency of high-resolution EM attacks using tight placements constraints as well as evaluating the security of LRPRFs implementations on smaller FPGA or ASIC technologies. Finally, a security evaluation of the spatial randomization countermeasure based on permutation networks could be performed using high-resolution EM attacks.

Bibliography

[1] Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. "Leakage-Resilient Symmetric Encryption via Re-keying". In: *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*. Ed. by Guido Bertoni and Jean-Sébastien Coron. Vol. 8086. Lecture Notes in Computer Science. Springer, 2013, pp. 471–488.

[2] Michel Abdalla and Mihir Bellare. "Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques". In: *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*. Ed. by Tatsuaki Okamoto. Vol. 1976. Lecture Notes in Computer Science. Springer, 2000, pp. 546–559.

[3] Dakshi Agrawal, Josyula R. Rao, and Pankaj Rohatgi. "Multi-channel Attacks". In: *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*. Ed. by Colin D. Walter, Çetin Kaya Koç, and Christof Paar. Vol. 2779. Lecture Notes in Computer Science. Springer, 2003, pp. 2–16.

[4] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. "Template Attacks in Principal Subspaces". In: *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*. Ed. by Louis Goubin and Mitsuru Matsui. Vol. 4249. Lecture Notes in Computer Science. Springer, 2006, pp. 1–14.

[5] Daniel Augot, Lejla Batina, Daniel J. Bernstein, Joppe Bos, Wouter Castryck Johannes Buchmann, Orr Dunkelman, Tim Güneysu, Shay Gueron, Tanja Lange Andreas Hülsing, Mohamed Saied Emam Mohamed, Christian Rechberger, Nicolas Sendrier Peter Schwabe, Frederik Vercauteren, and Bo-Yin Yang. *Initial recommendations of long-term secure post-quantum system*. `http://pqcrypto.eu.org/docs/initial-recommendations.pdf`. 2015.

[6] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. "The Sorcerer's Apprentice Guide to Fault Attacks". In: *Proceedings of the IEEE* 94.2 (2006), pp. 370–382.

[7] Alessandro Barenghi, Guido Bertoni, Luca Breveglieri, Mauro Pellicioli, and Gerardo Pelosi. "Low Voltage Fault Attacks to AES". In: *HOST 2010, Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 13-14 June 2010, Anaheim Convention Center, California, USA*. Ed. by Jim Plusquellic and Ken Mai. IEEE Computer Society, 2010, pp. 7–12.

[8] Alessandro Barenghi, Cédric Hocquet, David Bol, François-Xavier Standaert, Francesco Regazzoni, and Israel Koren. "Exploring the Feasibility of Low Cost Fault Injection Attacks on Sub-threshold Devices through an Example of a 65nm AES Implementation". In: *RFID. Security and Privacy - 7th International Workshop, RFIDSec 2011, Amherst, USA, June 26-28, 2011, Revised Selected Papers*. Ed. by Ari Juels and Christof Paar. Vol. 7055. Lecture Notes in Computer Science. Springer, 2011, pp. 48–60.

[9] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. "Mutual Information Analysis: a Comprehensive Study". In: *J. Cryptology* 24.2 (2011), pp. 269–291.

[10] Sonia Belaïd, Fabrizio De Santis, Johann Heyszl, Stefan Mangard, Marcel Medwed, Jörn-Marc Schmidt, François-Xavier Standaert, and Stefan Tillich. "Towards fresh re-keying with leakage-resilient PRFs: cipher design principles and analysis". In: *J. Cryptographic Engineering* 4.3 (2014), pp. 157–171.

[11] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. Ed. by Janos Simon. ACM, 1988, pp. 1–10.

[12] Ingrid Biehl, Bernd Meyer, and Volker Müller. "Differential Fault Attacks on Elliptic Curve Cryptosystems". In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*. Ed. by Mihir Bellare. Vol. 1880. Lecture Notes in Computer Science. Springer, 2000, pp. 131–146.

[13] Eli Biham and Adi Shamir. "Differential Cryptanalysis of DES-like Cryptosystems". In: *J. Cryptology* 4.1 (1991), pp. 3–72.

[14] Eli Biham and Adi Shamir. "Differential Fault Analysis of Secret Key Cryptosystems". In: *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*. Ed. by Burton S. Kaliski Jr. Vol. 1294. Lecture Notes in Computer Science. Springer, 1997, pp. 513–525.

[15] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. "A More Efficient AES Threshold Implementation". In: *Progress in Cryp-*

tology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings. Ed. by David Pointcheval and Damien Vergnaud. Vol. 8469. Lecture Notes in Computer Science. Springer, 2014, pp. 267–284.

[16] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. “Higher-Order Threshold Implementations”. In: *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. Lecture Notes in Computer Science. Springer, 2014, pp. 326–343.

[17] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, Natalia N. Tokareva, and Valeriya Vitkup. “Threshold implementations of small S-boxes”. In: *Cryptography and Communications* 7.1 (2015), pp. 3–33.

[18] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. “Cryptanalysis of a White Box AES Implementation”. In: *Selected Areas in Cryptography: 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*. Ed. by Helena Handschuh and M. Anwar Hasan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 227–240.

[19] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. “On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract)”. In: *Advances in Cryptology - EUROCRYPT ’97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*. Ed. by Walter Fumy. Vol. 1233. Lecture Notes in Computer Science. Springer, 1997, pp. 37–51.

[20] Joppe W. Bos, Charles Hubain, Wil Michiels, Cristofaro Mune, Eloi Sanfelix Gonzalez, and Philippe Teuwen. “White-Box Cryptography: Don’t Forget About Grey Box Attacks”. In: *IACR Cryptology ePrint Archive* (2017).

[21] Alfred Brauer. “On addition chains”. In: *Bull. Amer. Math. Soc.* 45.10 (Oct. 1939), pp. 736–739.

[22] Eric Brier, Christophe Clavier, and Francis Olivier. “Correlation Power Analysis with a Leakage Model”. In: *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*. Ed. by Marc Joye and Jean-Jacques Quisquater. Vol. 3156. Lecture Notes in Computer Science. Springer, 2004, pp. 16–29.

[23] Eric Brier, Helena Handschuh, and Christophe Tymen. “Fast Primitives for Internal Data Scrambling in Tamper Resistant Hardware”. In: *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*. Ed. by Çetin Kaya Koç, David Naccache,

and Christof Paar. Vol. 2162. Lecture Notes in Computer Science. Springer, 2001, pp. 16–27.

[24] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. "Towards Sound Approaches to Counteract Power-Analysis Attacks". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 398–412.

[25] Suresh Chari, Charanjit Jutla, Josyula R. Rao, and Pankaj Rohatgi. "A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards". In: *In* 2^{nd} *Advanced Encryption Standard (AES) Candidate Conference*. 1999.

[26] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. "Template Attacks". In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer, 2002, pp. 13–28.

[27] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *NISTIR 8105: Report on Post-Quantum Cryptography*. Tech. rep. NIST, Apr. 2016.

[28] Omar Choudary and Markus G. Kuhn. "Efficient Template Attacks". In: *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*. Ed. by Aurélien Francillon and Pankaj Rohatgi. Vol. 8419. Lecture Notes in Computer Science. Springer, 2013, pp. 253–270.

[29] Stanley Chow, Phil Eisen, Harold Johnson, and Paul C. van Oorschot. "A White-Box DES Implementation for DRM Applications". In: *Digital Rights Management: ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002. Revised Papers*. Ed. by Joan Feigenbaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 1–15.

[30] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. "Differential Power Analysis in the Presence of Hardware Countermeasures". In: *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1965. Lecture Notes in Computer Science. Springer, 2000, pp. 252–263.

[31] Jean-Sébastien Coron and Louis Goubin. "On Boolean and Arithmetic Masking against Differential Power Analysis". In: *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August*

17-18, 2000, Proceedings. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1965. Lecture Notes in Computer Science. Springer, 2000, pp. 231–237.

[32] Jean-Sébastien Coron and Ilya Kizhvatov. “An Efficient Method for Random Delay Generation in Embedded Software”. In: *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*. Ed. by Christophe Clavier and Kris Gaj. Vol. 5747. Lecture Notes in Computer Science. Springer, 2009, pp. 156–170.

[33] Joan Daemen and Vincent Rijmen. *AES Proposal: Rijndael*. 1999.

[34] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[35] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, Ventzislav Nikov, and Svetla Nikova. “Higher-Order Threshold Implementation of the AES S-Box”. In: *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*. Ed. by Naofumi Homma and Marcel Medwed. Vol. 9514. Lecture Notes in Computer Science. Springer, 2015, pp. 259–272.

[36] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, and Svetla Nikova. “Higher-Order Glitch Resistant Implementation of the PRESENT S-Box”. In: *Cryptography and Information Security in the Balkans - First International Conference, BalkanCryptSec 2014, Istanbul, Turkey, October 16-17, 2014, Revised Selected Papers*. Ed. by Berna Ors and Bart Preneel. Vol. 9024. Lecture Notes in Computer Science. Springer, 2014, pp. 75–93.

[37] Fabrizio De Santis, Tobias Bauer, and Georg Sigl. “Hiding Higher-Order Univariate Leakages by Shuffling Polynomial Masking Schemes: A More Efficient, Shuffled, and Higher-Order Masked AES S-box”. In: *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016*. Ed. by Begül Bilgin, Svetla Nikova, and Vincent Rijmen. ACM, 2016, pp. 17–26.

[38] Fabrizio De Santis, Tobias Bauer, and Georg Sigl. “Squeezing Polynomial Masking in Tower Fields - A Higher-Order Masked AES S-Box”. In: *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*. Ed. by Kerstin Lemke-Rust and Michael Tunstall. Vol. 10146. Lecture Notes in Computer Science. Springer, 2016, pp. 192–208.

[39] Fabrizio De Santis, Oscar M. Guillen, Ermin Sakic, and Georg Sigl. “Ciphertext-Only Fault Attacks on PRESENT”. In: *Lightweight Cryptography for Security and Privacy - Third International Workshop, LightSec 2014, Istanbul, Turkey, September*

1-2, 2014, Revised Selected Papers. Ed. by Thomas Eisenbarth and Erdinç Öztürk. Vol. 8898. Lecture Notes in Computer Science. Springer, 2014, pp. 85–108.

[40] Fabrizio De Santis and Stefan Rass. "On Efficient Leakage-Resilient Pseudorandom Functions with Hard-to-Invert Leakages". In: *Progress in Cryptology - LATINCRYPT 2014 - Third International Conference on Cryptology and Information Security in Latin America, Florianópolis, Brazil, September 17-19, 2014, Revised Selected Papers*. Ed. by Diego F. Aranha and Alfred Menezes. Vol. 8895. Lecture Notes in Computer Science. Springer, 2014, pp. 127–145.

[41] Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. "Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security". In: *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*. Ed. by Naofumi Homma and Marcel Medwed. Vol. 9514. Lecture Notes in Computer Science. Springer, 2015, pp. 225–241.

[42] Yevgeniy Dodis and Krzysztof Pietrzak. "Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks". In: *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*. Ed. by Tal Rabin. Vol. 6223. Lecture Notes in Computer Science. Springer, 2010, pp. 21–40.

[43] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. "Univariate side channel attacks and leakage modeling". In: *J. Cryptographic Engineering* 1.2 (2011), pp. 123–144.

[44] Richard Durstenfeld. "Algorithm 235: Random Permutation". In: ACM, 1964, p. 420.

[45] Stefan Dziembowski and Krzysztof Pietrzak. "Leakage-Resilient Cryptography". In: *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*. IEEE Computer Society, 2008, pp. 293–302.

[46] Eurosmart. *Eurosmart Confirms Overall Growth Trend For The Smart Security Industry Steady upward trajectory for worldwide secure element shipments in 2016 and 2017*. 2016.

[47] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. "Practical Leakage-Resilient Symmetric Cryptography". In: *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. Lecture Notes in Computer Science. Springer, 2012, pp. 213–232.

[48] Tse-Yun Feng. "A Survey of Interconnection Networks". In: *Computer* 14.12 (Dec. 1981), pp. 12–27.

[49] Julie Ferrigno and Martin Hlavác. “When AES blinks: introducing optical side channel”. In: *IET Information Security* 2.3 (2008), pp. 94–98.

[50] Wieland Fischer and Berndt M. Gammel. “Masking at Gate Level in the Presence of Glitches”. In: *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*. Ed. by Josyula R. Rao and Berk Sunar. Vol. 3659. Lecture Notes in Computer Science. Springer, 2005, pp. 187–200.

[51] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. “Fault Attacks on AES with Faulty Ciphertexts Only”. In: *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013*. Ed. by Wieland Fischer and Jörn-Marc Schmidt. IEEE Computer Society, 2013, pp. 108–118.

[52] Daniel Genkin, Adi Shamir, and Eran Tromer. “Acoustic Cryptanalysis”. In: *J. Cryptology* 30.2 (2017), pp. 392–443.

[53] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. “Mutual Information Analysis”. In: *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer, 2008, pp. 426–442.

[54] Commerce Secretary Announces New Standard for Global Information Security. 2001.

[55] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. “How to construct random functions”. In: *J. ACM* 33.4 (1986), pp. 792–807.

[56] Louis Goubin. “A Sound Method for Switching between Boolean and Arithmetic Masking”. In: *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*. Ed. by Çetin Kaya Koç, David Naccache, and Christof Paar. Vol. 2162. Lecture Notes in Computer Science. Springer, 2001, pp. 3–15.

[57] Louis Goubin and Jacques Patarin. “DES and Differential Power Analysis (The "Duplication" Method)”. In: *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES’99, Worcester, MA, USA, August 12-13, 1999, Proceedings*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1717. Lecture Notes in Computer Science. Springer, 1999, pp. 158–172.

[58] Hannes Groß, Stefan Mangard, and Thomas Korak. “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*. Ed. by Begül Bilgin, Svetla Nikova, and Vincent Rijmen. ACM, 2016, p. 3.

[59] Oscar M. Guillen, Michael Gruber, and Fabrizio De Santis. “Low-Cost Setup for Localized Semi-invasive Optical Fault Injection Attacks - How Low Can We Go?” In: *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*. Ed. by Sylvain Guilley. Vol. 10348. Lecture Notes in Computer Science. Springer, 2017, pp. 207–222.

[60] Johann Heyszl, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. “Localized Electromagnetic Analysis of Cryptographic Implementations”. In: *Topics in Cryptology - CT-RSA 2012 - The Cryptographers’ Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*. Ed. by Orr Dunkelman. Vol. 7178. Lecture Notes in Computer Science. Springer, 2012, pp. 231–244.

[61] Johann Heyszl, Dominik Merli, Benedikt Heinz, Fabrizio De Santis, and Georg Sigl. “Strengths and Limitations of High-Resolution Electromagnetic Field Measurements for Side-Channel Analysis”. In: *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers*. Ed. by Stefan Mangard. Vol. 7771. Lecture Notes in Computer Science. Springer, 2012, pp. 248–262.

[62] Naofumi Homma, Yu-ichi Hayashi, Noriyuki Miura, Daisuke Fujimoto, Daichi Tanaka, Makoto Nagata, and Takafumi Aoki. “EM Attack Is Non-invasive? - Design Methodology and Validity Verification of EM Attack Sensor”. In: *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. Lecture Notes in Computer Science. Springer, 2014, pp. 1–16.

[63] Michael Hutter and Jörn-Marc Schmidt. “The Temperature Side Channel and Heating Fault Attacks”. In: *IACR Cryptology ePrint Archive* (2014).

[64] Auguste Kerckhoffs. “La cryptographie militaire”. In: *Journal des sciences militaires* IX (Jan. 1883), pp. 5–83.

[65] HeeSeok Kim, Seokhie Hong, and Jongin Lim. “A Fast and Provably Secure Higher-Order Masking of AES S-Box”. In: *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*. Ed. by Bart Preneel and Tsuyoshi Takagi. Vol. 6917. Lecture Notes in Computer Science. Springer, 2011, pp. 95–107.

[66] Paul C. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. In: *Advances in Cryptology - CRYPTO ’96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22,*

1996, Proceedings. Ed. by Neal Koblitz. Vol. 1109. Lecture Notes in Computer Science. Springer, 1996, pp. 104–113.

[67] Paul C. Kocher. *Leak-resistant cryptographic indexed key update*. Patent. US 6539092. Mar. 2003.

[68] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. “Differential Power Analysis”. In: *Advances in Cryptology - CRYPTO ’99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 388–397.

[69] Sebastian Kutzner, Phuong Ha Nguyen, and Axel Poschmann. “Enabling 3-Share Threshold Implementations for all 4-Bit S-Boxes”. In: *Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers*. Ed. by Hyang-Sook Lee and Dong-Guk Han. Vol. 8565. Lecture Notes in Computer Science. Springer, 2013, pp. 91–108.

[70] Kerstin Lemke-Rust and Christof Paar. “Gaussian Mixture Models for Higher-Order Side Channel Analysis”. In: *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Vol. 4727. Lecture Notes in Computer Science. Springer, 2007, pp. 14–27.

[71] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[72] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. “Side-Channel Leakage of Masked CMOS Gates”. In: *Topics in Cryptology - CT-RSA 2005, The Cryptographers’ Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*. Ed. by Alfred Menezes. Vol. 3376. Lecture Notes in Computer Science. Springer, 2005, pp. 351–365.

[73] Mitsuru Matsui. “Linear Cryptanalysis Method for DES Cipher”. In: *Advances in Cryptology - EUROCRYPT ’93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*. Ed. by Tor Helleseth. Vol. 765. Lecture Notes in Computer Science. Springer, 1993, pp. 386–397.

[74] Marcel Medwed and François-Xavier Standaert. “Extractors against side-channel attacks: weak or strong?” In: *J. Cryptographic Engineering* 1.3 (2011), pp. 231–241.

[75] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. “Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices”. In: *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May*

3-6, 2010. Proceedings. Ed. by Daniel J. Bernstein and Tanja Lange. Vol. 6055. Lecture Notes in Computer Science. Springer, 2010, pp. 279–296.

[76] Marcel Medwed, François-Xavier Standaert, and Antoine Joux. "Towards Super-Exponential Side-Channel Security with Efficient Leakage-Resilient PRFs". In: *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. Lecture Notes in Computer Science. Springer, 2012, pp. 193–212.

[77] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[78] Nele Mentens, Benedikt Gierlichs, and Ingrid Verbauwhede. "Power and Fault Analysis Resistance in Hardware through Dynamic Reconfiguration". In: *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer, 2008, pp. 346–362.

[79] Amir Moradi, Mario Kirschbaum, Thomas Eisenbarth, and Christof Paar. "Masked Dual-Rail Precharge Logic Encounters State-of-the-Art Power Analysis Methods". In: *IEEE Trans. VLSI Syst.* 20.9 (2012), pp. 1578–1589.

[80] Amir Moradi and Oliver Mischke. "On the Simplicity of Converting Leakages from Multivariate to Univariate - (Case Study of a Glitch-Resistant Masking Scheme)". In: *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*. Ed. by Guido Bertoni and Jean-Sébastien Coron. Vol. 8086. Lecture Notes in Computer Science. Springer, 2013, pp. 1–20.

[81] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. "Correlation-Enhanced Power Analysis Collision Attack". In: *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*. Ed. by Stefan Mangard and François-Xavier Standaert. Vol. 6225. Lecture Notes in Computer Science. Springer, 2010, pp. 125–139.

[82] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. "Pushing the Limits: A Very Compact and a Threshold Implementation of AES". In: *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*. Ed. by Kenneth G. Paterson. Vol. 6632. Lecture Notes in Computer Science. Springer, 2011, pp. 69–88.

[83] Debdeep Mukhopadhyay. "An Improved Fault Based Attack of the Advanced Encryption Standard". In: *Progress in Cryptology - AFRICACRYPT 2009, Second*

International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings. Ed. by Bart Preneel. Vol. 5580. Lecture Notes in Computer Science. Springer, 2009, pp. 421–434.

[84] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. "Threshold Implementations Against Side-Channel Attacks and Glitches". In: *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*. Ed. by Peng Ning, Sihan Qing, and Ninghui Li. Vol. 4307. Lecture Notes in Computer Science. Springer, 2006, pp. 529–545.

[85] NIST Computer Security Division. *Advanced Encryption Standard (AES)*. FIPS Publication 197. National Institute of Standards and Technology, U.S. Department of Commerce, 2001.

[86] NIST Computer Security Division. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. FIPS Publication 202. National Institute of Standards and Technology, U.S. Department of Commerce, May 2014.

[87] Christof Paar and Jan Pelzl. *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer Berlin Heidelberg, 2009.

[88] Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. "A block cipher based pseudo random number generator secure against side-channel key recovery". In: *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*. Ed. by Masayuki Abe and Virgil D. Gligor. ACM, 2008, pp. 56–65.

[89] Krzysztof Pietrzak. "A Leakage-Resilient Mode of Operation". In: *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*. Ed. by Antoine Joux. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 462–482.

[90] Gilles Piret and Jean-Jacques Quisquater. "A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD". In: *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*. Ed. by Colin D. Walter, Çetin Kaya Koç, and Christof Paar. Vol. 2779. Lecture Notes in Computer Science. Springer, 2003, pp. 77–88.

[91] Thomas Popp, Mario Kirschbaum, Thomas Zefferer, and Stefan Mangard. "Evaluation of the Masked Logic Style MDPL on a Prototype Chip". In: *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*. Ed. by Pascal Paillier and Ingrid

Verbauwhede. Vol. 4727. Lecture Notes in Computer Science. Springer, 2007, pp. 81–94.

[92] François Poucheret, Lyonel Barthe, Pascal Benoit, Lionel Torres, Philippe Maurine, and Michel Robert. "Spatial EM jamming: A countermeasure against EM Analysis?" In: *18th IEEE/IFIP VLSI-SoC 2010, IEEE/IFIP WG 10.5 International Conference on Very Large Scale Integration of System-on-Chip, Madrid, Spain, 27-29 September 2010*. IEEE, 2010, pp. 105–110.

[93] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. "Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach". In: *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. Lecture Notes in Computer Science. Springer, 2016, pp. 61–81.

[94] Emmanuel Prouff and Matthieu Rivain. "Masking against Side-Channel Attacks: A Formal Security Proof". In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 142–159.

[95] Emmanuel Prouff and Thomas Roche. "Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols". In: *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*. Ed. by Bart Preneel and Tsuyoshi Takagi. Vol. 6917. Lecture Notes in Computer Science. Springer, 2011, pp. 63–78.

[96] Jean-Jacques Quisquater and David Samyde. "ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards". In: *Smart Card Programming and Security: International Conference on Research in Smart Cards, E-smart 2001 Cannes, France, September 19–21, 2001 Proceedings*. Ed. by Isabelle Attali and Thomas Jensen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 200–210.

[97] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. "Consolidating Masking Schemes". In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9215. Lecture Notes in Computer Science. Springer, 2015, pp. 764–783.

[98] Vincent Rijmen. *Efficient Implementation of the Rijndael S-box*.

[99] Matthieu Rivain and Emmanuel Prouff. “Provably Secure Higher-Order Masking of AES”. In: *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*. Ed. by Stefan Mangard and François-Xavier Standaert. Vol. 6225. Lecture Notes in Computer Science. Springer, 2010, pp. 413–427.

[100] Matthieu Rivain, Emmanuel Prouff, and Julien Doget. “Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers”. In: *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*. Ed. by Christophe Clavier and Kris Gaj. Vol. 5747. Lecture Notes in Computer Science. Springer, 2009, pp. 171–188.

[101] Thomas Roche and Emmanuel Prouff. “Higher-order glitch free implementation of the AES using Secure Multi-Party Computation protocols - Extended version”. In: *J. Cryptographic Engineering* 2.2 (2012), pp. 111–127.

[102] Atri Rudra, Pradeep K. Dubey, Charanjit S. Jutla, Vijay Kumar, Josyula R. Rao, and Pankaj Rohatgi. “Efficient Rijndael Encryption Implementation with Composite Field Arithmetic”. In: *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*. Ed. by Çetin Kaya Koç, David Naccache, and Christof Paar. Vol. 2162. Lecture Notes in Computer Science Generators. Springer, 2001, pp. 171–184.

[103] Dhiman Saha, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. “A Diagonal Fault Attack on the Advanced Encryption Standard”. In: *IACR Cryptology ePrint Archive* (2009).

[104] Pascal Sasdrich, Amir Moradi, and Tim Güneysu. “White-Box Cryptography in the Gray Box”. In: *Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*. Ed. by Thomas Peyrin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 185–203.

[105] Laurent Sauvage, Sylvain Guilley, and Yves Mathieu. “Electromagnetic Radiations of FPGAs: High Spatial Resolution Cartography and Attack on a Cryptographic Module”. In: *TRETS* 2.1 (2009), 4:1–4:24.

[106] Werner Schindler, Kerstin Lemke, and Christof Paar. “A Stochastic Model for Differential Side Channel Cryptanalysis”. In: *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*. Ed. by Josyula R. Rao and Berk Sunar. Vol. 3659. Lecture Notes in Computer Science. Springer, 2005, pp. 30–46.

[107] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. "Simple photonic emission analysis of AES". In: *J. Cryptographic Engineering* 3.1 (2013), pp. 3–15.

[108] Jörn-Marc Schmidt, Michael Hutter, and Thomas Plos. "Optical Fault Attacks on AES: A Threat in Violet". In: *Sixth International Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2009, Lausanne, Switzerland, 6 September 2009*. Ed. by Luca Breveglieri, Israel Koren, David Naccache, Elisabeth Oswald, and Jean-Pierre Seifert. IEEE Computer Society, 2009, pp. 13–22.

[109] Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. "A Collision-Attack on AES: Combining Side Channel- and Differential-Attack". In: *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*. Ed. by Marc Joye and Jean-Jacques Quisquater. Vol. 3156. Lecture Notes in Computer Science. Springer, 2004, pp. 163–175.

[110] Kai Schramm, Thomas J. Wollinger, and Christof Paar. "A New Class of Collision Attacks and Its Application to DES". In: *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*. Ed. by Thomas Johansson. Vol. 2887. Lecture Notes in Computer Science. Springer, 2003, pp. 206–222.

[111] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (1979), pp. 612–613.

[112] Claude E. Shannon. "Communication Theory of Secrecy Systems". In: *The Bell System Technical Journal* 28.4 (Oct. 1949), pp. 656–715.

[113] Joseph H. Silverman. "Fast Multiplication in Finite Fields GF(2^N)". In: *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1717. Lecture Notes in Computer Science. Springer, 1999, pp. 122–134.

[114] Sergei Skorobogatov. "Flash Memory 'Bumping' Attacks". In: *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*. Ed. by Stefan Mangard and François-Xavier Standaert. Vol. 6225. Lecture Notes in Computer Science. Springer, 2010, pp. 158–172.

[115] François-Xavier Standaert and Cédric Archambeau. "Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages". In: *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008.*

Proceedings. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer, 2008, pp. 411–425.

[116] François-Xavier Standaert, Tal Malkin, and Moti Yung. “A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks”. In: *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*. Ed. by Antoine Joux. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 443–461.

[117] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. “Leakage Resilient Cryptography in Practice”. In: *Towards Hardware-Intrinsic Security - Foundations and Practice*. Ed. by Ahmad-Reza Sadeghi and David Naccache. Information Security and Cryptography. Springer, 2010, pp. 99–134.

[118] Kris Tiri and Ingrid Verbauwhede. “A Dynamic and Differential CMOS Logic Style to Resist Power and Timing Attacks on Security IC’s”. In: *IACR Cryptology ePrint Archive* (2004).

[119] Kris Tiri and Ingrid Verbauwhede. “A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation”. In: *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*. IEEE Computer Society, 2004, pp. 246–251.

[120] Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. “Cryptanalysis of DES Implemented on Computers with Cache”. In: *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*. Ed. by Colin D. Walter, Çetin Kaya Koç, and Christof Paar. Vol. 2779. Lecture Notes in Computer Science. Springer, 2003, pp. 62–76.

[121] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. “An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks”. In: *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*. Ed. by Lars R. Knudsen and Huapeng Wu. Vol. 7707. Lecture Notes in Computer Science. Springer, 2012, pp. 390–406.

[122] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. “An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks”. In: *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*. Ed. by Lars R. Knudsen and Huapeng Wu. Vol. 7707. Lecture Notes in Computer Science. Springer, 2012, pp. 390–406.

[123] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. "Security Evaluations beyond Computing Power". In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 126–141.

[124] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. "Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note". In: *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. Lecture Notes in Computer Science. Springer, 2012, pp. 740–757.

[125] Nicolas Veyrat-Charvillon and François-Xavier Standaert. "Generic Side-Channel Distinguishers: Improvements and Limitations". In: *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*. Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 354–372.

[126] Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger. "An ASIC Implementation of the AES SBoxes". In: *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*. Ed. by Bart Preneel. Vol. 2271. Lecture Notes in Computer Science. Springer, 2002, pp. 67–78.

[127] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. "Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings". In: *Selected Areas in Cryptography: 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*. Ed. by Carlisle Adams, Ali Miri, and Michael Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 264–277.

[128] Yu Yu and François-Xavier Standaert. "Practical Leakage-Resilient Pseudorandom Objects with Minimum Public Randomness". In: *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco, CA, USA, February 25-March 1, 2013. Proceedings*. Ed. by Ed Dawson. Vol. 7779. Lecture Notes in Computer Science. Springer, 2013, pp. 223–238.

[129] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. "Practical leakage-resilient pseudorandom generators". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois,*

USA, October 4-8, 2010. Ed. by Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov. ACM, 2010, pp. 141–151.

List of Publications

Refereed Proceedings

[P20] Fabrizio De Santis, Andreas Schauer, and Georg Sigl. “ChaCha20-Poly1305 authenticated encryption for high-speed embedded IoT applications”. In: *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*. Ed. by David Atienza and Giorgio Di Natale. IEEE, 2017, pp. 692–697.

[P19] Oscar M. Guillen, Michael Gruber, and Fabrizio De Santis. “Low-Cost Setup for Localized Semi-invasive Optical Fault Injection Attacks - How Low Can We Go?” In: *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*. Ed. by Sylvain Guilley. Vol. 10348. Lecture Notes in Computer Science. Springer, 2017, pp. 207–222.

[P18] Philipp Koppermann, Fabrizio De Santis, Johann Heyszl, and Georg Sigl. “Automatic generation of high-performance modular multipliers for arbitrary mersenne primes on FPGAs”. In: *2017 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2017, McLean, VA, USA, May 1-5, 2017*. IEEE Computer Society, 2017, pp. 35–40.

[P17] Hermann Seuschek, Fabrizio De Santis, and Oscar M. Guillen. “Side-channel leakage aware instruction scheduling”. In: *Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems, CS2@HiPEAC 2017, Stockholm, Sweden, January 24, 2017*. Ed. by Mats Brorsson, Zhonghai Lu, Giovanni Agosta, Alessandro Barenghi, and Gerardo Pelosi. ACM, 2017, pp. 7–12.

[P16] Florian Unterstein, Johann Heyszl, Fabrizio De Santis, and Robert Specht. “Dissecting Leakage Resilient PRFs with Multivariate Localized EM Attacks - A Practical Security Evaluation on FPGA”. In: *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*. Ed. by Sylvain Guilley. Vol. 10348. Lecture Notes in Computer Science. Springer, 2017, pp. 34–49.

[P15] Fabrizio De Santis, Tobias Bauer, and Georg Sigl. “Hiding Higher-Order Univariate Leakages by Shuffling Polynomial Masking Schemes: A More Efficient, Shuffled, and Higher-Order Masked AES S-box”. In: *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016*. Ed. by Begül Bilgin, Svetla Nikova, and Vincent Rijmen. ACM, 2016, pp. 17–26.

[P14] Fabrizio De Santis, Tobias Bauer, and Georg Sigl. “Squeezing Polynomial Masking in Tower Fields - A Higher-Order Masked AES S-Box”. In: *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*. Ed. by Kerstin Lemke-Rust and Michael Tunstall. Vol. 10146. Lecture Notes in Computer Science. Springer, 2016, pp. 192–208.

[P13] Oscar M. Guillen, Fabrizio De Santis, Ralf Brederlow, and Georg Sigl. “Towards Side-Channel Secure Firmware Updates - A Minimalist Anomaly Detection Approach”. In: *Foundations and Practice of Security - 9th International Symposium, FPS 2016, Québec City, QC, Canada, October 24-25, 2016, Revised Selected Papers*. Ed. by Frédéric Cuppens, Lingyu Wang, Nora Cuppens-Boulahia, Nadia Tawbi, and Joaquín García-Alfaro. Vol. 10128. Lecture Notes in Computer Science. Springer, 2016, pp. 345–360.

[P12] Philipp Koppermann, Fabrizio De Santis, Johann Heyszl, and Georg Sigl. “X25519 Hardware Implementation for Low-Latency Applications”. In: *2016 Euromicro Conference on Digital System Design, DSD 2016, Limassol, Cyprus, August 31 - September 2, 2016*. Ed. by Paris Kitsos. IEEE Computer Society, 2016, pp. 99–106.

[P11] Hermann Seuschek, Johann Heyszl, and Fabrizio De Santis. “A Cautionary Note: Side-Channel Leakage Implications of Deterministic Signature Schemes”. In: *Proceedings of the Third Workshop on Cryptography and Security in Computing Systems, CS2@HiPEAC, Prague, Czech Republic, January 20, 2016*. Ed. by Martin Palkovic, Giovanni Agosta, Alessandro Barenghi, Israel Koren, and Gerardo Pelosi. ACM, 2016, pp. 7–12.

[P10] Michael Tempelmeier, Fabrizio De Santis, Jens-Peter Kaps, and Georg Sigl. “An area-optimized serial implementation of ICEPOLE authenticated encryption schemes”. In: *2016 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016, McLean, VA, USA, May 3-5, 2016*. Ed. by William H. Robinson, Swarup Bhunia, and Ryan Kastner. IEEE Computer Society, 2016, pp. 49–54.

[P9] Fabrizio De Santis, Oscar M. Guillen, Ermin Sakic, and Georg Sigl. “Ciphertext-Only Fault Attacks on PRESENT”. In: *Lightweight Cryptography for Security and*

Privacy - Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014, Revised Selected Papers. Ed. by Thomas Eisenbarth and Erdinç Öztürk. Vol. 8898. Lecture Notes in Computer Science. Springer, 2014, pp. 85–108.

[P8] Fabrizio De Santis and Stefan Rass. "On Efficient Leakage-Resilient Pseudorandom Functions with Hard-to-Invert Leakages". In: *Progress in Cryptology - LATINCRYPT 2014 - Third International Conference on Cryptology and Information Security in Latin America, Florianópolis, Brazil, September 17-19, 2014, Revised Selected Papers*. Ed. by Diego F. Aranha and Alfred Menezes. Vol. 8895. Lecture Notes in Computer Science. Springer, 2014, pp. 127–145.

[P7] Fabrizio De Santis, Michael Kasper, Stefan Mangard, Georg Sigl, Oliver Stein, and Marc Stöttinger. "On the Relationship between Correlation Power Analysis and the Stochastic Approach: An ASIC Designer Perspective". In: *Progress in Cryptology - INDOCRYPT 2013 - 14th International Conference on Cryptology in India, Mumbai, India, December 7-10, 2013. Proceedings*. Ed. by Goutam Paul and Serge Vaudenay. Vol. 8250. Lecture Notes in Computer Science. Springer, 2013, pp. 215–226.

[P6] Johann Heyszl, Andreas Ibing, Stefan Mangard, Fabrizio De Santis, and Georg Sigl. "Clustering Algorithms for Non-profiled Single-Execution Attacks on Exponentiations". In: *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*. Ed. by Aurélien Francillon and Pankaj Rohatgi. Vol. 8419. Lecture Notes in Computer Science. Springer, 2013, pp. 79–93.

[P5] Johann Heyszl, Dominik Merli, Benedikt Heinz, Fabrizio De Santis, and Georg Sigl. "Strengths and Limitations of High-Resolution Electromagnetic Field Measurements for Side-Channel Analysis". In: *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers*. Ed. by Stefan Mangard. Vol. 7771. Lecture Notes in Computer Science. Springer, 2012, pp. 248–262.

[P4] Matthias Hiller, Fabrizio De Santis, Dominik Merli, and Georg Sigl. "Reliability bound and channel capacity of IBS-based fuzzy embedders". In: *2012 NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2012, Erlangen, Germany, June 25-28, 2012*. Ed. by Umeshkumar D. Patel, Khaled Benkrid, and David Merodio. IEEE, 2012, pp. 213–220.

[P3] Alessandro Barenghi, Guido Bertoni, Fabrizio De Santis, and Filippo Melzani. "On the Efficiency of Design Time Evaluation of the Resistance to Power Attacks". In: *14th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, DSD 2011, August 31 - September 2, 2011, Oulu, Finland*. IEEE Computer Society, 2011, pp. 777–785.

[P2] Giovanni Agosta, Alessandro Barenghi, Fabrizio De Santis, and Gerardo Pelosi. “Record Setting Software Implementation of DES Using CUDA”. In: *Seventh International Conference on Information Technology: New Generations, ITNG 2010, Las Vegas, Nevada, USA, 12-14 April 2010*. Ed. by Shahram Latifi. IEEE Computer Society, 2010, pp. 748–755.

[P1] Giovanni Agosta, Alessandro Barenghi, Fabrizio De Santis, Andrea Di Biagio, and Gerardo Pelosi. “Fast Disk Encryption through GPGPU Acceleration”. In: *2009 International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2009, Higashi Hiroshima, Japan, 8-11 December 2009*. IEEE Computer Society, 2009, pp. 102–109.

Refereed Journals

[J4] Philipp Koppermann, Fabrizio De Santis, Johann Heyszl, and Georg Sigl. “Fast FPGA Implementations of Diffie-Hellman on the Kummer Surface of a Genus-2 Curve”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst. (TCHES)* 2018.1 (2018), pp. 1–17.

[J3] Philipp Koppermann, Fabrizio De Santis, Johann Heyszl, and Georg Sigl. “Low-latency X25519 hardware implementation: breaking the 100 microseconds barrier”. In: *Microprocessors and Microsystems - Embedded Hardware Design* 52 (2017), pp. 491–497.

[J2] Silvan Streit and Fabrizio De Santis. “Post-Quantum Key Exchange on ARMv8-A: A New Hope for NEON Made Simple”. In: *IEEE Transactions on Computers* (2017).

[J1] Sonia Belaïd, Fabrizio De Santis, Johann Heyszl, Stefan Mangard, Marcel Medwed, Jörn-Marc Schmidt, François-Xavier Standaert, and Stefan Tillich. “Towards fresh re-keying with leakage-resilient PRFs: cipher design principles and analysis”. In: *J. Cryptographic Engineering* 4.3 (2014), pp. 157–171.

Refereed Book Chapters

[B1] Alessandro Barenghi, Luca Breveglieri, Fabrizio De Santis, Filippo Melzani, Andrea Palomba, and Gerardo Pelosi. “Design Time Engineering of Side Channel Resistant Cipher Implementations”. In: *Theory and Practice of Cryptography Solutions for Secure Information Systems*. Ed. by Atilla Elçi, Mehmet A. Orgun, Josef Pieprzyk, Alexander G. Chefranov, Huaxiong Wang, and Rajan Shankaran.

Advances in Information Security, Privacy, and Ethics (AISPE). IGI Global, 2013, pp. 133–157.