Interactive Side-Channel Analysis

Kostas Papagiannopoulos

Radboud University

A thesis submitted for the degree of

Doctor of Philosophy

2020

ISBN: 978-94-6380-791-3

NUR: 919

Typeset using LaTeX

Cover design by: Vaggelis Margaritis

Printed by: Proefschriftmaken NL

Institute for Computing and Information Sciences
Radboud University

Netherlands Organisation for Scientific Research

The work in this thesis is funded by the Netherlands Organisation for Scientific Research NWO through project PROFIL. The author was employed by the Institute for Computing and Information Sciences, Radboud University, Nijmegen.

Interactive Side-Channel Analysis

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus prof. dr. J.H.J.M. van Krieken,
volgens besluit van het college van decanen
in het openbaar te verdedigen op dinsdag 26 Mei 2020
om 16.30 uur precies

door

Konstantinos Papagiannopoulos

geboren op 7 maart 1988
te Ioannina, Griekenland

Promotor:

Prof. dr. Lejla Batina

Manuscriptcommissie:

Prof. dr. Elena Marchiori
Dr. Annelie Heuser - (IRISA, Rennes, Frankrijk)
Dr. Johann Heyszl - (Fraunhofer-Gesellschaft, München, Duitsland)
Prof. dr. Stefan Mangard - (Technische Universität Graz, Oostenrijk)
Prof. dr. François-Xavier Standaert - (Université catholique de Louvain, België)

Interactive Side-Channel Analysis

DOCTORAL THESIS

to obtain the degree of doctor
from Radboud University Nijmegen
on the authority of the Rector Magnificus prof. dr. J.H.J.M. van Krieken,
according to the decision of the Council of Deans
to be defended in public on Tuesday, May 26, 2020
at 16.30 hours

by

Konstantinos Papagiannopoulos

born on March 7, 1988
In Ioannina, Greece

Promotor:

Prof. dr. Lejla Batina

Doctoral Thesis Committee:

Prof. dr. Elena Marchiori
Dr. Annelie Heuser - (IRISA, Rennes, France)
Dr. Johann Heyszl - (Fraunhofer-Gesellschaft, München, Germany)
Prof. dr. Stefan Mangard - (TU Graz, Austria)
Prof. dr. François-Xavier Standaert - (Université catholique de Louvain, Belgium)

To Apostolis, Miranta and Ophelia, for all these years in Ioannina.

To Maria, for all our years in Athens and Nijmegen.

Acknowledgements

First and foremost, I would like to thank my parents Apostolis Papagiannopoulos and Miranta Tzima for helping me throughout my entire life, supporting my choices and providing all imaginable help for 31 years. Teaching me mathematics and history was just the beginning of my study life and a memory I will always cherish upon. Likewise, I would like to thank my sister Ophelia Papagiannopoulou for standing close to me and making my life joyful.

In the same spirit, I would like to thank Maria Panagiotou for standing by my side during my entire adult life, making me happy, motivated and courageous. I also want to thank all my friends in Nijmegen and Athens that believed in me throughout these years.

I need to provide special thanks to my supervisor Lejla Batina for supporting the full path towards a PhD in cryptography and side-channel analysis. In addition, I would like to thank Erik Poll, Bart Jacobs, Irma Haerkens, Peter Schwabe, Gergely Alpar, Sander Uijlen, Anna Kransova, Joost Renes, Joost Rijneveld and the whole ICIS department for creating a friendly atmosphere and a pleasant work environment.

The works that constitute this thesis were carried out after lengthy discussions, arguments and collaborations with several people that I am proud to call coworkers. Thus, I want to thank to my coauthors Lukasz Chmielewski, Niels Samwel, Louiza Papachristodoulou, Vasilios Mavroudis, Ko Stoffelen, Liran Lerman, Nikita Veshchikov, Antonio de La Piedra, Erik Schneider, Wouter de Groot Apostolos Fournaris, Thierry Simon, Joan Daemen, Vincent Grosso, Pedro Maat Costa Massolino, Francesco Regazzoni, Benjamin Gregoire, Niek Timmers, Pol Van Aubel, Christian Doerr, Christos Andrikos, Giorgos Rassias, Guilherme Perin and Alberto Sonnino.

Last but not least I would like to thank all the students that I was lucky enough to meet during the courses of hardware security, cryptographic engineering and BSc/MSc project supervision. Thanks for giving me the strength to become better so that I can hopefully meet your expectations. Thus, special thanks to Niels Samwell, Ko Stoffelen and Thierry Simon, all of which started as students in ICIS and turned into full-time coworkers. Thanks also to Brian Detourbet, Philipp Jakubeit, Erik Schneider, Wouter de Groot, Tim van Dijk, Mael Virgoroux and many others that were involved in side-channel research in our workspace.

Abstract

The modern, always-online world relies on numerous electronic devices. Ensuring the unobstructed operation of electronic transactions is quintessential yet non-trivial to achieve. Many devices operate in particularly resource-constraint environments, often trying to achieve security with very narrow margins.

Physical attacks such as side-channel cryptanalysis and fault injection pose a serious threat against the security of embedded devices. Techniques such as Differential Power Analysis and Template Attacks exploit the power consumption and electromagnetic emission of embedded targets, compromising cryptography in otherwise secure mathematical ciphers.

To meet the security needs of our society, numerous countermeasures have been deployed against such attacks. Masking and shuffling rank among the most popular choices, yet they do not come for free. Very often the cost of deploying them effectively makes the implementation cost prohibitive, leading to situations where only partially secure products are used in the field. To address these issues, this thesis puts forward the following three contribution points.

First, it develops efficient masking and shuffling countermeasures. To improve the performance of these countermeasures it relies on high speed assembly-based implementations that push the limit of ARM and AVR devices. At the same time it investigates closely the actual security level of such devices, aiming to remove leakage effects that hinder their full deployment.

Second, this thesis works toward a closer understanding of the various countermeasures against physical attacks. Instead of viewing countermeasures as isolated components, it promotes a holistic approach that examines the interactions between countermeasures, security and performance of a secure cryptographic implementation. Through information-theoretic analysis, we establish the tradeoff between randomness and masking/shuffling countermeasures, culminating in Reduced Randomness Masking/Shuffling schemes. In the same spirit, we link the fault injection resistance of duplicated ciphers, infective protection and build-in fault detection to the side-channel security of a device. The tradeoffs established can assist the countermeasure designer prior deployment and result in effective, yet affordable security.

Third, this thesis integrates new attack vectors to the existing arsenal of the side-channel adversary. It provides a close inspection of location-based attacks on ARM devices and assesses their real-world impact. Concurrently, it takes steps towards modeling location leakage, aiming to understand its root cause and once again to establish tradeoffs between attack parameters and attack impact.

Samenvatting

De moderne wereld die altijd online is, vertrouwt op vele elektronische apparaten. Het is gebruikelijk dat elektronische transacties onbelemmerd uitgevoerd worden. Echter is dat niet vanzelfsprekend. Veel apparaten functioneren met beperkte middelen, waardoor ze vaak minimaal worden beveiligd.

Fysieke aanvallen zoals side-channel cryptoanalyse en fault injection zijn een serieuze dreiging voor de beveiliging van ingebouwde apparaten. Technieken zoals Differential Power Analysis en Template Attacks maken gebruik van het stroomverbruik of elektromagnetische emissie van de ingebouwde apparaten. Hierdoor wordt cryptografie die wiskundig veilig is onbetrouwbaar.

Om te voldoen aan de hedendaagse behoefte aan beveiliging van elektronische apparaten zijn er verschillende maatregelen ingezet tegen zulke aanvallen. Maskeren en rangschikken behoren tot de meest populaire maatregelen, toch zijn deze niet zonder kosten. Vaak zijn deze kosten zo hoog dat het niet uitvoerbaar is om ze volledig in te zetten. Er ontstaan dan situaties waarin producten worden gebruikt die slechts voor een deel zijn beveiligd. Om deze problemen aan te pakken, komen in dit proefschrift de volgende drie bijdragen aan bod.

Ten eerste worden in dit proefschrift efficiënte maatregelen voor maskeren en rangschikken ontwikkeld. Er worden op assembly gebaseerde implementaties gebruikt die de grenzen van ARM en AVR apparaten verleggen. Daarnaast belemmeren beveiligingslekkages de maatregelen. Daarom wordt tegelijkertijd het beveiligingsniveau van de apparaten onderzocht met als doel om de lekkages te voorkomen.

Ten tweede wordt in dit proefschrift toegewerkt naar een beter inzicht in de verschillende maatregelen tegen fysieke aanvallen. In plaats van de maatregelen als geïsoleerde componenten te bekijken, wordt met een integrale benadering gekeken naar de interacties tussen de maatregelen, de beveiliging en de prestaties van een beveiligde cryptografische implementatie. Door informatie-theoretische analyse stellen we een afweging vast enerzijds tussen de kosten van willekeurige getallen en anderzijds maatregelen als maskeren/rangschikken. Dit geeft "Reduced Randomness Masking/Shuffling" als resultaat. In dezelfde trant voegen we de weerstand tegen fault injection van gedupliceerd versleutelingsalgoritmes, infective bescherming en ingebouwde detectie van fault injection samen onder de side-channel beveiliging van een apparaat. De vastgestelde afwegingen helpen bij het ontwikkelen van nieuwe maatregelen voordat ze worden toegepast. Dit resulteert in een beveiliging die effectief en kosten efficiënt is.

Ten slotte integreert dit proefschrift nieuwe aanvalsvectoren in het bestaande arsenaal van side-channel aanvallen. Het levert een nauwkeurige inspectie van op locatie gebaseerde aanvallen op ARM apparaten en beoordeelt het daadwerkelijke effect. Tegelijkertijd worden er stappen gezet naar het modelleren van op locatie gebaseerde lekkages. Dit heeft als doel een beter inzicht te geven in de voornaamste oorzaak van deze lekkages. Bovendien worden afwegingen vastgesteld tussen de aanvalsparameters en het aanvalseffect.

Abbreviations

AES Advanced Encryption Standard
BBP Belaïd, Benhamouda, Passelègue
CBC Cipher Block Chaining mode
CNN Convolutive Neural Network
CPA Correlation Power Analysis
DPA Differential Power Analysis
ECC Elliptic Curve Cryptography
EM ElectroMagnetic
FI Fault Injection
HD Hamming Distance
HDL Hardware Description Language
HI Hypothetical Information
HO Higher Order
HW Hamming Weight
ID Instruction Duplication
IT Information Theoretic
ILA Independent Leakage Assumption
ISW Ishai-Sahai-Wagner
LDA Linear Discriminant Analysis
LUT LookUp Table
NI Non-Interfering
NN Neural Network
MI Mutual Information
MIA Mutual Information Analysis
ML Machine Learning
MLP Multi Layer Perceptron
PCA Principal Component Analysis
PI Perceived Infromation
RSA Rivest-Shamir-Adleman
RNG Random Number Generation
RRM Reduced Randomness Masking
RRS Reduced Randomness Shuffling
SASCA Soft Analytical Side-Channel Attack
SCA Side-Channel Analysis
SIMD Single Instruction Multiple Data
SNI Strong Non-Interfering
SNR Signal to Noise Ratio
SR Success Rate
SSL Secure Socket Layer
TA Template Attacks
TI Threshold Implementation
TRNG True Random Number Generation
TVLA Test Vector Leakage Assessment

Contents

List of Figures

Chapter 1

Introduction

"We don't really know what electricity is, we merely know how to use it."

Few names fit our modern era more aptly than the "age of information". Historically, information exchange, processing and storage are deeply embroidered in humanity's societal fabric and emerge in all conceivable communication means, ranging form the oral tradition and collective consciousness of a society to a written alphabet and its literary derivatives. More interestingly though, the 20th century, with the advent of information technology, telecommunications and electronics, has emancipated humanity from several historic barriers in information exchange, processing and storage. Nowadays, information is unshackled by the limitations of human nature such as proximity, limited storage capacity and erroneous memory. As a result, our modern, always-interconnected society can daily deploy a huge number of digital transactions to communicate, operate and thrive.

The "steam engine" driving this new wave of modernization is no other than the infamous Internet of Things. Enabled by the low cost of electronics and the wide-spread availability of networks, the Internet of Things is the process of enhancing everyday objects such as identities, passports, transport cards, cellphones with internet connectivity, data-processing and storage capabilities. Several industrial sectors, including construction, logistics, agriculture, energy production/distribution and automotive manufacturing, are already harnessing the transformative impact of IoT on their production lines. The emerging paradigm is a globally connected world, consisting of billions of device nodes. Resembling an intra-cerebral architecture with billions of "brain cells", every node communicates and interacts through thousands of neural connections, powering our planet's cortex.

However, close to every life-changing human invention lies the price of progress. History has recorded several occasions where inventions caused rippling waves in human society. For instance, the ancient story of the Tower of Babel vividly portraits the invention of human languages, as well as its impact. A huge tower (or possibly a metaphor for human progress and ambition) is constructed and attempts to reach the sky, only to crumble under its own weight. Allegorically speaking, the story describes how the invention of language had spectacular effects, yet it also caused confusion,

miscommunication and isolation between people talking in different languages. Fast-forward to the modern era, the internet of things and the emerging online world had huge positive impacts in our society. The price of progress in our era largely pertains to the human security and privacy of this brave new world. The billions of interconnected IoT devices contain our personal data and transactions, that is, our digital identity and fingerprint. Should an attacker gain unauthorized access to our electronic life, he use it to expose our personal communications, to fraudulently imitate us in the online world or to simply deny us access to online services. More importantly, modern business models rely heavily on surveillance capitalism methods that closely monitor user preferences, map routes, purchase history in a constant struggle to accumulate behavior surplus [225]. Thus the end user is under constant pressure by vested interest to reveal personal information and make it subject to data rendition. To avoid another tower collapse we need to make sure that our multi-faceted online identity remains secure and safe from prying eyes.

Our ally in this effort is the science of cryptography. History offers an abundance of examples where cryptographically protecting secrets was crucial to geopolitical interests, as seen in the ancient world and the Ceasar cipher [1], in the middle-ages and the Da Vinci devices [2] and even in modern warfare with Enigma [3]. Through the ages, the core principles of cryptography remained immutable. A message sender and a message receiver want to communicate securely over an insecure channel and keep their communication secret from adversaries. In an era were billions of devices constantly communicate over insecure channels, the need for efficient and effective cryptography is even more dire.

The mathematical foundations of cryptography rely on on computationally hard problems. Most encryption techniques constitute of an algorithm that receives an input, known as plaintext and generates an output, known as ciphertext. Cryptanalysis studies such algorithms, trying to invert the encryption process and recover the plaintext from the ciphertext. To this end, various tools are being deployed, including linear and differential cryptanalysis [33, 143], impossible differentials [121], integral cryptanalysis [67], slide attacks [35] and others. Every technique works under the common "black box" assumption, that is the adversary can observe the input and output of a cryptographic algorithm. In other words, the internal algorithm states are fully opaque and the adversary is not privy to any inside information.

As mentioned, every cryptographic algorithm, also known as cipher, begins its existence as an abstract mathematical structure that is meant to be cryptanalyzed. Thorough work in the the "black box" model ensures that the cipher is adequately protected for such usage. The next natural step in the cipher's lifecycle is a real-world implementation. The cipher can get deployed in any electronic device, ranging from supercomputing clusters, to laptops, cellphones, even smartcards and RFID tags. Transforming a purely mathematical object to an actual implementation however comes with a price: every device has physical characteristics that are now observable by the adversary. The execution time, the power consumption, the electromagnetic emission and other features of a device are new tools for the adversary which allow him a blurry yet potent view in the internal states of the cipher. All these physical characteristics form a conglomerate of attack vectors that gives birth to side-channel

analysis and the so-called "grey box" model.

Before attempting to bind the concept of side-channel analysis to a formal definition, we opt to attach a more aesthetic feeling to it, using various examples. In general, human intellect is often struggling to capture directly the truth behind the laws of the physical world. Interestingly, we excel much better at learning through interactions, as any interaction in the physical world leaves a trace behind. Learning and understanding through interaction is a very common scientific pattern. For example, Michael Faraday developed his understanding of electromagnetism by constructing inductive electric motors. Confirming the quote in the beginning of this chapter, the underlying physical phenomenon (electromagnetic induction) was unclear. However, Faraday managed to aptly understand it by a fitting experimental interaction. That is, an interaction produced a *side-channel* that shed light upon the laws of physics. Naturally, interactive experiments do not capture the laws of physics in their entirety, yet they spearhead an ongoing process towards understanding. Sometimes discovering the necessary interaction can be very hard. For instance, the properties of neutrinos remained elusive for a long time, largely because of the particle's weak interaction with matter. Physicists constructed neutrino detectors by deploying huge water tanks that are able to capture the Cherenkov radiation emitted when an incoming neutrino creates an electron or muon in the water. Through this strange side-channel neutrinos were understood much better, prompting more interactive experiments. Not surprisingly, natural life is full of interactions and side-channels. Human beings can comprehend emotion and empathy indirectly, often through subtle hints, facial expression and body posture. Flora and fauna are unable to keep track of calendar days, yet a slight increase in temperature, a different breeze of air or an early sunrise are side-channels telling them that spring is coming.

Going back to side-channel analysis, we provide the following definition:

Side-channel analysis is the art of uncovering the truth through interactions

The definition is fairly generic and resembles the research methods of most natural sciences. In our case, the "truth" is the hidden internal state of a cipher. The side-channels that we use stem from the way ciphers interact with physical devices. Time, electrical power, electromagnetic/photonic emission, sound and others are simply mediums that convey information about such interactions. They byproduct of such interaction is often referred to as "side-channel leakage".

1.1 Motivation & Research Questions

This thesis is titled "Interactive Side-Channel Analysis". On a surface level, its goal is to exploit the plethora of interactions between mathematical ciphers and the physical world, hence the adjective "interactive". Any successful interaction results in an insecure implementation and prompts research towards side-channel countermeasures,

i.e. methods that eliminate such interactions. The majority of novel side-channel research tries to identify new unexpected interactions (new attacks) and to fully utilize existing interactions (maximized leakage exploitation). Ultimately, it tries to make such interactions very hard to exploit, resulting in secure devices (side-channel countermeasures). This thesis provides advances in all these three directions.

However, this is not the sole reason behind the "interactive" titling. Side-channel analysis and countermeasures are not only the byproducts of interaction. Instead, they actively interact with other security components such as Random Number Generation and Fault Injection protection. Even more, they showcase internal interactions, i.e. different types of side-channels such as data and location channels interact between themselves. Thus, on a deeper level the goal of the thesis is to also demonstrate the interactions between side-channel analysis and other crucial parts of implementation and security. This thesis demonstrates and analyzes such interactions aiming to understand hardware security in a holistic sense, to provide efficient and effective countermeasures and to offer a plethora of design options to the engineers of secure devices.

So far, several attacks make use of side-channel interactions, including timing attacks [29], Differential Power Analysis [125], Template Attacks [50], Soft-Analytical Side-Channel Attacks [215], Neural Network attacks [138] and others. Concurrently multiple countermeasures such as masking [49], shuffling [216], special logic styles [99], clock jitter and leakage-resilient cryptography [199] have attempted to thwart side-channel leakage problems. Motivating our study, we observe that side-channel attacks have been widely deployed against real products, such as the Mifare product used (among others) in the Dutch OV-chipkaart [4, 193], doors that use electronic lock mechanisms [83], Intel processors [123] the Secure Socket Layer network protocol [42], Wolf SSL [182], even allegedly secured memory units [14]. At the same time, countermeasures are deployed in Android and Apple smartphones to protect the NFC payment systems, debit and credit cards use masking and shuffling to prevent cloning and Ledger wallets employ protection to prevent unauthorized access to bitcoins.

Having established the core points of Interactive Side-Channel Analysis, we proceed to describe the specific research questions behind this work.

1. **Side-channel countermeasure implementation.** One of the most widely-spread countermeasure against side-channel attacks is the masking countermeasure, considered for either software or hardware (threshold implementations). Starting from multi-party computation, masking is applied on the algorithmic layer of a cipher and is capable of randomizing the intermediate states of its computation. Although several masking schemes have been proposed, higher-order masking is still perceived by the semiconductor industry as a potent yet costly countermeasure. Thus, the interaction between masking countermeasures and implementation cost leads to the following research question:

   Can we implement efficient high-speed masking schemes in modern microcontroller units?

2. **Countermeasures and the physical layer.** Masking countermeasures against

side-channel analysis are usually developed with a theoretical model in mind. However, the often unpredictable electrical characteristics of the physical layer do not strictly adhere to the model, causing masking to underperform. Thus, the interaction between masking countermeasures and the physical layer cost leads to the following research question:

Can we understand and eliminate the impact of the physical layer on deployed masking countermeasures?

3. **Side-channel analysis and random number generation.** Countermeasures against SCA are inherently reliant on random number generation in order to provide protection. This often involves a costly overhead for implementation that can render countermeasures inefficient. Exploring ways to "recycle" existing randomness can assist towards cheaper solutions. Thus, the interaction between side-channel analysis and random number generation leads to the following research questions:

   Can we use randomness more effectively when protecting ciphers? Can we establish the tradeoff between side-channel resistance and RNG overhead?

4. **Side-channel and fault injection resistance.** Along with side-channel analysis, another large class of attacks relies on injecting fault during a cipher computation. Protecting against this class requires often redundant computations, which in turn can enhance existing side-channel leakage. Protecting against both SCA and FI attack classes is non-trivial since we need to balance between opposing forces. Thus, the interaction between side-channel analysis and fault injection countermeasures leads to the following research question:

   Can we establish the tradeoff between side-channel and fault-injection resistance?

5. **Data and location side-channels.** Hardware security research is constantly trying to discover unknown vulnerabilities. Location-based attacks are more recent side-channel vulnerabilities and originate from the fact that chip structures such as memory and register file emit distinctive information when accessed. The interactions resulting in location leakage as well as the interaction between standard data-based and location-based leakage lead to the following research questions:

   Can we successfully exploit location leakage in modern devices? Can we analyze data and location leakage jointly?

All the above research questions culminate in the following general question:

Can we effectively understand the interactive links between side-channel analysis, implementations, the physical layer, random number generation, fault injection and location-based attacks? Ultimately, can we use these interactions to improve security?

1.2 Thesis Organization

Chapter 2 provides an overview of basic profiled and non-profiled side-channel attacks such as correlation power analysis and template attacks. In addition, it provides an introduction to the masking and shuffling countermeasures. The chapter also describes basic metrics used in side-channel analysis, including success rate and information-theoretic approaches. Last, it discusses horizontal side-channel attacks, which is the common denominator in most analyses carried out in this thesis. Moreover, it practically demonstrates the necessity for such attacks by examining specific scenarios.

Chapter 3 improves the current state of the art by creating an efficient 2nd-order masked implementation of PRESENT, showing that midrange ARM-based architectures (Cortex-M) can host masked implementations efficiently. In addition, it demonstrates how the NEON vector unit on high-end ARM Cortex-A8 processors can be used to obtain efficient masked AES implementations. Finally, the chapter attempts to bridge the gap between theory and practice in masking countermeasures in a twofold manner. First, it demonstrates potential out-of-model leakages in ARM Cortex-M and Cortex-A architectures. Continuing, it identifies similar effects in the ATmega163 and attempts to mitigate such problems by crafts the first (to our knowledge) 1st-order masked implementation that is capable of resisting 1st-order, univariate attacks.

Contribution of the author: Focusing on several microcontrollers, the author has developed high-performance cryptographic implementations of secure ciphers, merging the concepts of bitslicing and algebraic decompositions together with masking countermeasures. In addition, the author has performed power analysis attacks against the developed implementations, analyzing the security level and root-causing the side-channel leakage effects. The results of this effort are included in "Bitsliced Masking and ARM: Friends or Foes?" [70], "Vectorizing Higher-Order Masking" [94] and "Mind the Gap: Towards Secure 1st-order Masking in Software" [161].

Chapter 4 describes low-randomness versions of standard masking and shuffling countermeasures, which we refer to as Recycled Randomness Masking and Reduced Randomness Shuffling respectively. The novel countermeasures reduce the RNG overhead and establish a direct link between the randomness cost and the noisy leakage security level provided by a countermeasure.

Contribution of the author: The author established the theoretical links between randomness and masking/shuffling side-channel countermeasures. In addition, he puts forward both a noise-based information-theoretic analysis and formal verification methods for improved countermeasures. The results of this effort are included in "Low Randomness Masking and Shuffling: An Evaluation Using Mutual Information" [160].

Chapter 5 examines the tradeoff between fault injection resistance and side-channel analysis resistance by investigating fault injection countermeasures and new cipher structures with build-in fault injection protection. Using an information-theoretic approach and the Hidden Markov Model, it examines n-plication and infective countermeasures. Moreover, the chapter introduces FRIET, a fault-resistant cipher design and employs a Soft-Analytical Side-Channel Attack to evaluate the impact of FI resistance to side-channel analysis.

Contribution of the author: The author's contribution is performing a theoretical and practical side-channel analysis of duplication and infection countermeasures. In addition, the author has analyzed the side-channel leakage of FRIET using soft-analytical side-channel attacks, linking it to fault resistance. The results of this effort are included in "Instruction Duplication: Leaky and Not Too Fault-Tolerant!" [59] and "Towards Lightweight Cryptographic Primitives with Built-in Fault-Detection" [190].

Chapter 6 investigates the fairly unexplored location-based leakage in the context of modern microcontrollers. It provides simple spatial model that partially captures the effect of location-based leakages and uses it to simulate different theoretical scenarios that enhance or diminish location-based leakage. Continuing, it carries out the first practical location-based attack on the SRAM of a modern ARM Cortex-M4, using standard side-channel techniques, as well as neural network classifiers. Finally, it investigate the recently proposed countermeasure of Boolean exponent splitting from the perspective of location-based attacks. It present for the first time a hybrid attack, where data leakage is combined with location leakage and analyze the countermeasure's effectiveness against it.

Contribution of the author: Motivated by experimental observations, the author has developed the theoretical model for location-based leakage. In addition he has performed information-theoretic analyses and template-based attacks on various real and simulated leakage scenarios. The results of this effort are included in "Location, location, location: Revisiting modeling and exploitation for location-based side channel leakages" [7], "Location-based leakages: New directions in modeling and exploiting" [8] and in "Boolean Exponent Splitting" [209].

Chapter 7 summarizes our conclusions and provides an open discussion about the future research directions on side-channel attacks and countermeasures.

Chapter 2

Preliminaries

2.1 Notation

This thesis uses the following concepts and notation to describe cryptanalytic side-channel analysis experiments, quantities and any other related information.

- *Symmetric cipher:* is a bijective function parameterized by a secret key K, an input plaintext In, computing a ciphertext C. Inverting the function is computationally infeasible without the knowledge of the secret key K. Random variables are denoted with capital letters and we use them to describe such discrete quantities. Instances of random variables and constant values are denoted with lowercase letters, i.e. k, in, c. The domain of these discrete random variables is denoted using calligraphic letters, i.e. $\mathcal{K}, \mathcal{I}, \mathcal{C}$. Naturally the domain contains all possible values of these variables.

- *Advanced Encryption Standard*: AES-128 is a symmetric cipher operates on a single block of data known as the state. Each AES round is composed of 4 operations: AddRoundKey, Sbox, ShiftRows and MixColumns [69].

- *PRESENT and RECTANGLE cipher:* Lightweight symmetric ciphers operating on 64-bit blocks of data. [36, 224]. Both include substitution and permutation operations in every round.

- *Side-channel experiment:* the experimental process under which an analyst acquires side-channel data information by observing the timing, power supply, electromagnetic emmission or any other covert channels. The analysis goal is usually to learn information about a key-dependent intermediate variable, often denoted as V.

- *Traces:* the side channel observables. Each trace is a vector consisting of multiple time points and usually corresponds to a single cipher execution. Traces are denoted with capital bold letters that are used for random variable vectors and matrices. Variable $\mathbf{L}$ denotes the leakage of a trace and has domain $\mathcal{L}$ which usually is $(-\infty, \infty)$. Initially all traces are analog signals. The most common

approach to process and analyze them is a digitization process, where the analog signal is converted to its digital equivalent. Typically, we use the Nyquist rate, thus we digitize at a sampling rate (samples/sec) that is equal to at least twice the core frequency of our signal.

- *Samples:* the discretized points in time of a trace. Samples are denoted with subscript notation in a trace, i.e. $l_1, l_2, \ldots, l_{1000}$ for a trace with 1000 samples.

- Leakage function: the assumed model that defines the side-channel behavior of a key-dependent intermediate variable. The leakage function can follow the identity model, the Hamming weight, the Hamming distance model or any other custom behavior.

- *Statistical distributions:* the specified random variables follow certain statistical distributions. This is denoted as $X \sim Distribution$. The notation $Unif(\{a, b\})$, $Bern(p)$, $Binomial(n, p)$ and $\mathcal{N}(\mu, \sigma^2)$ denotes random variables with uniform, Bernoulli, binomial and normal probability distributions respectively. Parameter p denotes the probability of Bernoulli/binomial trials and μ, σ^2, denote the mean and variance of the normal distribution. The set $\{a, b\}$ denotes that the discrete uniform distribution can receive value a or b equiprobably. The notation $E[\cdot]$, $Var[\cdot]$ and $H[\cdot]$ describes the expected value, variance and entropy of a random variable. Notation $Pr(\cdot)$ implies the probability of an event/random variable.

- *Side-channel attack:* The analysis process during which the attacker tries to identify the correct key via side-channel experiments. Typically, it involves the process of distinguishing between different key guesses.

- *Success rate:* Typically, a side-channel attack attack is deemed successful if the side-channel distinguisher ranks the correct key first among all key guesses. Expanding this, one can also define the rank of the correct key i.e. after the distinguisher ranking, he can observe how far away is the correct key from the top position.

2.2 From Standard to Horizontal Attacks

There exist several ways to classify the plethora of side-channel attacks and techniques. Common classification options consider whether the adversary has access or not to an open device and therefore partition SCA to profiled and unprofiled attacks. Other classification attempts split techniques based on the distinguisher they use, often separating between standard statistical distinguishers (comparison-based or partitioned-based) and more advance distinguishers based on information theory, machine learning and deep learning. Likewise, classifications split techniques based on the nature of the side-channel used, resulting in timing [29, 124], power [125], electromagnetic [105], photonic [184] and others.

This thesis is closely linked to the concept of *horizontal* side-channel attacks therefore it necessitates a more modern classification attempt. Ergo, we aim to

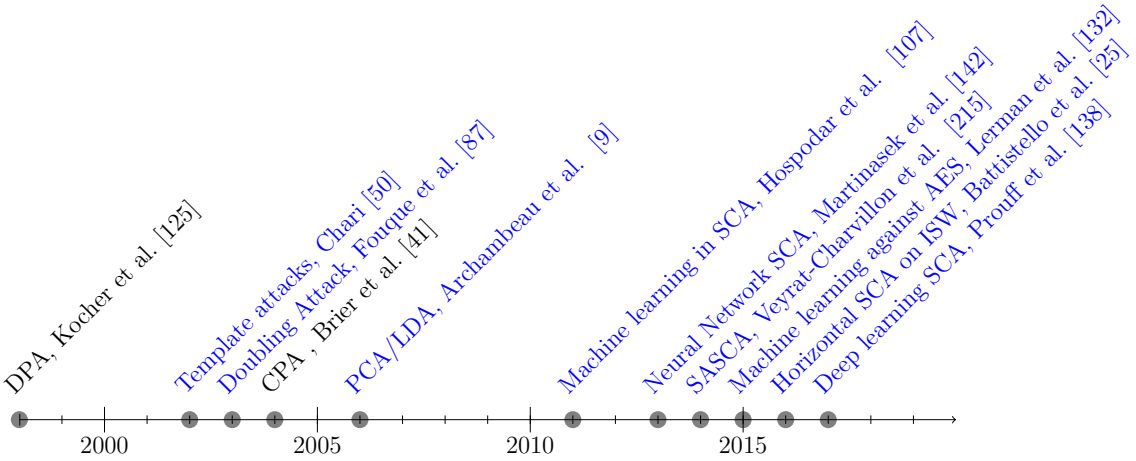


Figure 2.1: Standard (in black) and horizontal (in blue) attacks over time.

present the spectrum of side-channel techniques from the viewpoint of horizontality. Horizontality can be simply described as the effort to maximize the information extracted from a side-channel trace. In fact, ranking techniques based on horizontality is following closely the evolution of side-channel analysis over the years, i.e. it demonstrates the progress of the whole side-channel community in extracting as much leakage as possible and improving the attack capabilities of such cryptanalysis.

Researcher's have implicitly used various concepts of horizontality to attack public-key cryptography. Fouque et al. [87] and later Homma et al. [106] came up with techniques that performed side-channel cryptanalysis exploiting the noisy leakage *throughout* the whole execution of a scalar multiplication or modular exponentiation. The term 'horizontal attack' first appeared in the work of Clavier et al. [56], becoming the first conscious attempt to exploit this dimension. Through horizontal attacks they were able to exploit *multiple intermediate values*, increasing the effectiveness of their attack. In a concurrent fashion, researchers implicitly used horizontality to delve deeper into the leakage of a device. Instead of relying on a single sample for their attack, Chari [50] opted for multivariate leakage models that exploit *multiple trace samples*, increasing once again the effectiveness of the attack. The majority of horizontal techniques can be viewed as an extension that aims to model multiple samples and combine multiple intermediate values. A timeline of standard (in black) and horizontal (in blue) side-channel attacks is visible in Figure 2.1. We continue with a short description of the horizontal and non-horizontal techniques that are relevant to this work.

Differential Power Analysis. DPA is the earliest side-channel attack and the first to link the instantaneous power consumption with the intermediate values of any cryptographic algorithm [125]. It utilizes a simple leakage model, namely that 0-valued bits in the cryptographic implementation leak differently compared to 1-valued bits. DPA is a univariate technique that typically targets a single intermediate value of the cipher. Therefore it can be classified as a non-horizontal attack.

Technique description. Let a key dependent intermediate value represented by

random variable V. DPA distinguishes the leakage of the device L when the value V is 0 or 1 i.e. we observe $(L|V = 0)$ and $(L|V = 1)$. Given input I and key guess K_g, the adversary computes a predicted intermediate value V_g. in For instance, in the case of AES, $V_g = Sbox(I \oplus K_g))$, where $Sbox()$ is the typical 8×8 AES Sbox.

Subsequently, the traceset L is split in two sets: one where $V_g = 0$ and one where $V_g = 1$. The adversary computes the difference of means between the 2 sets, namely δ. For the correct key guess (and thus a correct set splitting), δ increases with the number of traces, while for incorrect key guess it tends to small values.

$$\delta = E[L|V_g = 0] - E[L|V_g = 1] \tag{2.1}$$

Correlation Power Analysis. Being one the most popular side-channel distinguishers, CPA [41] was among the first to introduce the concept of a leakage model and link the power consumption of the device to the Hamming weight of a single intermediate value or to the Hamming distance between intermediate values. CPA is an instance of DPA, which although being univariate, it tends to exploit more information since it can model the leakage of a full device register/bus instead of a single bit of that register/bus.

Technique description. In its first version, CPA relates the leakage of the device L with the Hamming weight of the intermediate value computed or Hamming distance between intermediate values, i.e. $HW(V)$ or $HD(V_1, V_2) = HW(V_1 \oplus V_2)$. Given input I and key guess K_g, the adversary computes a predicted intermediate value V_g and in the Hamming weight case, computes the $HW(V_g)$. Subsequently it compares the traceset L with the prediction using Pearson correlation $\rho_k(L, HW(V_g))$ and ranks the keys based on that. The correlation formula follows below.

$$\rho_k(L, HW(V_g)) = \frac{cov[L, HW(V_g)]}{\sqrt{Var[L] \cdot Var[HW(V_g)]}} \tag{2.2}$$

TVLA Methodology. The leakage detection methodology [60] is not a direct attack technique, yet it is also very often deployed in evaluations and even considered a requirement for certain security certification processes. The technique prioritizes leakage detection over leakage exploitation, speeding up evaluation. Although multivariate TVLA techniques exist, their adoption remains still at a nascent stage, therefore TVLA is a non-hotizontal technique.

Technique description. In detail, TVLA employs the random vs. fixed, non-specific, 1st-order t-test. That is, first it perform a random vs. fixed acquisition and obtain two distinct tracesets S_{fixed} and S_{random}, under the same encryption key. The input plaintext for S_{fixed} is set to a fixed value, while for S_{random}, the input is uniformly random. The implementation receives the fixed or random plaintext in a non-deterministic and randomly-interleaved way, as recommended by Schneider et al.

[185]. Following the data acquisition, the 1st-order t-test will assess whether the two sets S_{fixed}, S_{random} stem from the same population, using the following statistical test. Parameters μ_x and σ_x^2 are the estimated mean and variance of set x; n and m denote sizes of sets S_{fixed} and S_{random} respectively. The null hypothesis H_{null} is rejected at a given level of significance α (often set to 0.99999), if $|w| > t_{\alpha/2,\upsilon}$, where $t_{\alpha/2,\upsilon}$ is the value of the Student t distribution with υ degrees of freedom. In the evaluation context, rejecting H_{null} implies leakage detection, i.e. potential evidence of an insecure device.

$$\begin{aligned} H_{null}: &\quad \mu_{fixed} = \mu_{random} \\ H_{alt}: &\quad \mu_{fixed} \neq \mu_{random} \end{aligned} \tag{2.3}$$

$$w = \frac{\mu_{fixed} - \mu_{random}}{\sqrt{\frac{\sigma^2_{fixed}}{n} + \frac{\sigma^2_{random}}{m}}} \qquad \upsilon = \frac{(\frac{\sigma^2_{fixed}}{n} + \frac{\sigma^2_{random}}{m})^2}{\frac{\sigma^4_{fixed}}{n^2(n-1)} + \frac{\sigma^4_{random}}{m^2(m-1)}} \tag{2.4}$$

Template and Machine Learning Attacks. In many ways, template attacks put forward an early version of horizontal side-channels. The first version of TA uses a more complex leakage model (multivariate normal distribution), i.e. $L \sim \mathcal{N}(\mu, \Sigma)$, although any probability distribution is elegible for such an attack. To perform a TA, the adversary needs to select one or multiple sample points of the traceset where leakage is detected. The so-called points of interest are used during template training part of the mean vector (μ) and covariance matrix Σ.

TAs mark a divergence from the standard univariate attack scenario and the expansion towards more informative leakage models. Extracting the more key information from the leaky traces, lies essentially in the modeling capabilities of the template attack. A large amount of work is dedicated towards identifying good POIs and compressing the time samples such that the template can be build with a reasonable amount of data [9, 147].

In the same spirit, the advent of linear regression [183], improved dimensionality reduction methods [24, 145] and modern machine learning techniques [107, 132] often manages to increase the horizontal factor and improve the success rate of our attack or reduce the data complexity during the training phase. Such enhanced statistical techniques enable stronger attacks such as Online Template Attacks by Batina et al. [22], which manages to connect standard templates to horizontal attacks, since it uses TAs to identify horizontally same-data computations and attack ECC in a single-trace. Currently, the horizontal trend is still vibrant, since neural networks and deep learning techniques are widely used in order to detect POIs, improve attacks and extract more information via more sophisticated leakage models.

Technique description. Template attacks, including any machine learning or deep learning extensions, follow the same attack path. First they make use of an open device in order to profile certain instructions or cryptographic implementations,

i.e. the capture a training dataset. Subsequently they use this training dataset to create a model. Standard template attacks rely on a probabilistic model whose parameters are estimated using the training dataset. In other words, TA will estimate the statistical distributions $\hat{Pr}(\mathbf{L} = \mathbf{l}|K = k)$ for all $k \in \mathcal{K}$. Other techniques may use different approaches such as random forest training, neural network weight estimation etc. Once the training is complete, templates use a maximum likelihood distinguisher in order to match an unknown leakage to one of the known profiles. Assuming unknown leakage vector $\mathbf{l}$ and key $K$with domain $\mathcal{K}$, the TA will compute $Pr(\mathbf{L} = \mathbf{l}|K = k)$ for all $k \in \mathcal{K}$ and select the key candidate k with the highest probability. Other techniques may produce scores that reflect the same process. The probability computation follows below and is a direct application of the Bayes rule. In practice the probability computation gets replaced by the logarithm of the probability which simplifies computation and can solve numerical issues in higher dimensions [54].

$$Pr(k|\mathbf{l}) = \frac{Pr(\mathbf{l}|k)}{\sum_{k^* \in \mathcal{K}} Pr(\mathbf{L} = \mathbf{l}|K = k^*)} \tag{2.5}$$

Soft Analytical Attacks. Among the strongest horizontal attacks are the soft-analytical attacks proposed by Veyrat-Charvillon et al. [215]. SASCA improves our exploitation capabilities by considering more than a single intermediate value to attack. Instead, it observes several leaky intermediates $V_1, \ldots, V_m$ and the corresponding leakage $L_1, \ldots, L_m$. Subsequently it constructs a graph that combines all such information while taking into account the exact implementation of the cipher. Several variants of SASCA exist, usually attempting to balance the effort between training an informative model and keeping the training complexity reasonable.

Attack description. SASCA can be viewed as an enhanced template attack. That is, in the first step all intermediate values $V_1, \ldots, V_m$ are profiled using templating techniques. In the second step a factor graph is constructed, modeling the entirety of relations between such values. Establishing the link between the intermediates is enhancing the side-channel attack with extra observables: partially leaky intermediates may not be informative enough on their own, yet once combined they improve the attack's success rate. This combination is performed via the Belief Propagation algorithm [126] which iterates through the graph in order to inform nodes about the behavior of their neighbors.

2.2.1 Information-Theoretic Framework

A core approach in side-channel evaluations is the set of several information-theoretic measures that have been developed to assist the evaluator and provide insight into the quantity and nature of the side-channel leakage. In this subsection we describe the three core information theoretic metrics that are encountered throughout most chapters. Namely we discuss the features and computation methods with regards to Mutual Information, Perceived Information and Hypothetical Information.

Mutual Information. The core information theoretic metric is mutual information [197]. $MI(\mathbf{L};V)$ quantifies the amount of information that the leakage $\mathbf{L}$ conveys about the key-dependent intermediate V. To compute MI we need knowledge of the true distribution of the leakage, i.e. we implicitly assume that we are aware of the exact leakage model without any need for template building or any form of distribution estimation. Naturally, this applies to theoretical scenarios, where the evaluator attempts to gauge the impact of certain leakage models with a specific case in mind. MI computations essentially implies perfect modeling capabilities and perfect matching between the built model and the leakage of the target during the attack phase.

$$MI(\mathbf{L};V) = H[V] + \sum_{v\in\mathcal{V}} Pr[v] \cdot \int\limits_{\mathbf{l}\in\mathcal{L}^n} Pr[\mathbf{l}|v] \cdot \log_2 Pr[v|\mathbf{l}]\ \mathrm{d}\mathbf{l},$$
$$\text{where } Pr[v|\mathbf{l}] = \frac{Pr[\mathbf{l}|v]}{\sum_{v^*\in\mathcal{V}} Pr[\mathbf{l}|v^*]} \tag{2.6}$$

.

Perceived Information. To turn MI into a more flexible metric, Renauld et al. [178] put forward a version that captures the imperfections of a real-world experiment. PI alters MI in the two following ways. First, PI relies on the estimated statistical distribution of the leakage (instead of the true distribution), i.e. it factors in any possible modeling errors that are caused by template building with limited sampling/datasets. At the same time PI compares the estimated distribution against the true leakage distribution of the device-under-test, that is against the sampled leakage obtained from the device. Thus, it can also factor in assumption errors that are caused when template building has incorrect modeling assumptions, leading to imperfections that reduce the success rate of the side-channel technique. Unlike MI, PI can take negative values when the templating process has errors, demonstrating that the attack is incapable of key recovery.

$$PI(\mathbf{L};V) = H[R] - H_{true,model}[\mathbf{L}|V] = H[V] + \sum_{v\in\mathcal{V}} Pr[v] \int\limits_{\mathbf{l}\in\mathcal{L}^n} Pr_{true}[\mathbf{l}|v] \cdot log_2 Pr_{model}[v|\mathbf{l}]\ d\mathbf{l}$$
$$\text{where } Pr_{model}[v|\mathbf{l}] = \frac{Pr_{model}[\mathbf{l}|v]}{\sum_{v^*\in\mathcal{V}} Pr_{model}[\mathbf{l}|v^*]},\ Pr_{true}[\mathbf{l}|v] = \frac{1}{n_{test}}, n_{test} \text{ test set size} \tag{2.7}$$

Hypothetical Information. Hypothetical information bears similarities to both PI and MI. Like PI, HI is a real-world metric that relies on estimated statistical models of the leakage. Unlike PI, HI compares the estimated model against itself, i.e. it the amount of information that would be revealed by hypothetical data following the model distribution [115]. Like MI, HI has a non-negative value, so it can be viewed as it's real-world counterpart.

$$HI(\mathbf{L};V) = H[V] + \sum_{v\in\mathcal{V}} Pr[v] \cdot \int_{l\in\mathcal{L}} \hat{Pr}_{model}[l|v] \cdot log_2 \hat{Pr}_{model}[v|l] \;\; dl,$$

$$\text{where } \hat{Pr}_{model}[v|l] = \frac{\hat{Pr}_{model}[l|v]}{\sum_{v^*\in\mathcal{S}} \hat{Pr}_{model}[l|v^*]} \tag{2.8}$$

2.3 Countermeasures Against Side-Channels

Throughout this work we concentrate our efforts towards the two most important types of protection against side-channel attacks: *masking* and *shuffling*. These two techniques, in conjunction with noise, are able to provide security to a cryptographic implementation and demand stronger adversarial techniques to bypass. Both can be implemented on the algorithmic layer of cryptography. Thus, they do not rely on specialized electrical/electronic components and can be easily applied both in software and in hardware.

2.3.1 Masking

Chari et al., Goubin et al. and Messerges [49, 90, 148] were the first to suggest randomizing intermediate values with a secret sharing scheme, forcing the adversary to analyze higher-order statistical moments. Stemming from these, research has developed numerous masking schemes that use various algebraic operations to split the secret value into shares. Common choices for sharing the secret include multiplication, modular addition and affine transformations and others.

The most prevalent choice is no other than Boolean masking, predominantly popularized due its strong performance numbers. Analytically, a dth-order secure Boolean masking scheme splits a sensitive value x into d + 1 shares $(x_0, x_1, ..., x_d)$, as shown below.

$$x = x_0 \oplus x_1 \oplus \cdots \oplus x_d$$

The shares $(x_0, x_1, ..., x_d)$ are also referred to as the (d+ 1)-family of shares corresponding to x [180]. In a given $(d + 1)$-tuple of intermediate values, we let random variable S be the sensitive (key-dependent) intermediate value under attack and let random variables $M_0, \ldots, M_{d-1}$ be the masks used to protect the sensitive value. The leakage of a $(d + 1)$-tuple is described using the following random vector: $\mathbf{L} = (L_{S\oplus\bigoplus_{i=0}^{d-1} M_i}, L_{M_0}, \ldots, L_{M_{d-1}}) + \mathbf{N}$, where $L_{S\oplus\bigoplus_{i=0}^{d-1} M_i} = \mathsf{L_{id}}\ (S\oplus M_0\oplus\cdots\oplus M_d)$, $L_{M_i} = \mathsf{L_{id}}\ (M_i)$, $0 \le i \le d-1$ and $\mathbf{N}$ is a $(d+1)$-dimensional random vector representing Gaussian noise. We assume independent and equal noise σ^2 in every sample of the tuple.

Assuming sufficient noise and a specific leakage function, it has been shown that the number of traces required for a successful attack grows exponentially w.r.t. the security order d [49], i.e. masking performs noise amplification. Analytically, assuming that the masking shares leak independently and follow normal distribution $\mathcal{N}(\mu, \sigma^2)$,

Chari et al. [49] demonstrated that at the number of traces N required to distinguish with probability p between distributions $(\mathbf{L}|S=0)$ and $\mathbf{L}|S=1)$ satisfies the following inequality.

$$N \geq \sigma^{(d+4logp/log\sigma)}$$

This core result has solidified the resistance of masking schemes against side-channel attacks. Retrieving information about the secret becomes exponentially harder rendering it popular choices to countermeasure designers and implementors. Corroborating this core security notion, several additional security definitions have been used to specify the formal security properties of a masking scheme, and we revisit the most relevant below.

- *Probing-secure scheme.* We refer to a scheme that uses certain families of shares as $t-$probing-secure iff any set of at most t intermediate variables is independent from the sensitive values [111].
- *Non-interfering scheme.* We refer to a scheme as $t-$non-interfering ($t-$NI) iff any set of at most t intermediate variables can be perfectly simulated with at most t shares of each input [19].
- *Strongly non-interfering scheme.* We refer to a scheme as strong non-interfering ($t-$SNI) iff any set of at most t intermediate variables, where t_1 are on the internal variables and t_2 on the output variables, can be perfectly simulated with at most t_1 shares of each input [19].

2.3.2 Shuffling

The shuffling countermeasure results in spreading information over n different points in time, according to a random permutation $\mathbf{P}_n$ [181]. The permutation $\mathbf{P}_n$ is defined as a vector $(P_0, \ldots, P_n)$, where P_i represents the new position of element i and thus $\mathbf{P}_n$ is defined over the set of all possible n-dimensional permutations $\mathcal{P}_n$. For instance, assume two independent variables $\mathbf{X} = (X_0, X_1)$ that leak $\mathbf{L} = (L_{X_0}, L_{X_1})$ at different points in time. The shuffling scheme will generate a 2-dimensional permutation $\mathbf{P}_2$ s.t. $L_{X_0} = \mathsf{L}_{id}(X_{P_0}) + noise$ and $L_{X_1} = \mathsf{L}_{id}(X_{P_1}) + noise$. Charvillon et al. [216] have analyzed the security provided by shuffling, in addition to investigating several implementation techniques. We will refer to a permutation that shuffles n independent operations of a specific cipher layer as $\mathbf{P}_n^{\{o_1,\ldots,o_n\}}$

Similar to masking, applying the shuffling countermeasure implies a non-negligible randomness cost. Specifically, generating a permutation for shuffling k independent operations of the same type requires $k * \lceil log_2(k) \rceil$ random bits, using a slightly-biased version of the Knuth shuffle algorithm [122, 216]. In a practical scenario, shuffling only 16 AES Sboxes requires 640 random bits in total[1]. In order to deal with this RNG overhead, previous work on the shuffling countermeasure opted to reduce the amount

[1]The cipher runs for 10 rounds, permuting 16 independent operations of the same type (Sbox) per round. Every permutation requires $16 * \lceil log_2(16) \rceil$ random bits.

of possible permutations (random start index), to shuffle only in selected rounds (partial shuffling) or to use non-homogeneous shuffle patterns, where the amount of possible permutations varied between cipher layers [103, 181].

2.4 Three Cases in Favor of Horizontal Attacks

The contributions of this thesis are largely possibly due to the strong potential and improved exploitation capabilities offered by horizontal techniques. In Chapters 3, 4, 5 and 6 several attack and defense mechanisms are examined through the lens of horizontality, in order to provide close-to-optimal leakage analysis and in order to establish a strong, yet realistic adversarial model.

However, the side-channel literature has put a large amount of effort in univariate attacks, to the extent that side-channel analysis is largely considered a synonym of CPA. The relative ease, the straightforward application and unprofiled nature of CPA made it the de facto technique for side-channel analysis. On the contrary, template attacks (and as a result soft-analytical attacks) prompt more lengthy POI investigations and often involve less straightforward leakage models. Moreover, the profiled nature of templates and SASCA give rise to profile transferability issues [53]. All these technical difficulties often result in analysts questioning their effectiveness and applicability.

This section attempts to put an end to the reluctance against such techniques by discussing 3 real-world scenarios where horizontality plays a major role in deducing the correct result. More importantly, in all scenarios, constraining oneself to univariate techniques can result in partially correct or even misleading results. All three cases involve in some way side-channel information, yet not all three are limited to the standard cryptographic scenario of key-recovery through leaky observables.

2.4.1 Confusion coefficient, transparency order and cipher structure as a countermeasure

The first scenario under discussion is set in the classic side-channel context of leaky ciphers and key recovery. In particular, the literature contains several works have attempted to model the side-channel resistance of a cryptographic implementation. Prouff [169] and Fei et al. [86] were among the first to model the side-channel properties of cryptographic components, resulting in metrics such as the transparency order of a Boolean function and the confusion coefficient of a structure. A natural next step of the community was to try identifying structures and functions that possessed good side-channel resistance properties. The need for low-cost resistant implementations led several research lines towards modified 8×8 and 4×4 Sboxes [165,166]. Such structures were similar to existing Sboxes regarding linear and differential cryptanalysis, yet demonstrated improved numbers regarding the confusion coefficient.

Testing these Sboxes in theory and practice confirmed such intuition. Should an adversary opt to perform CPA on the Sbox output of a cipher (as is very often the

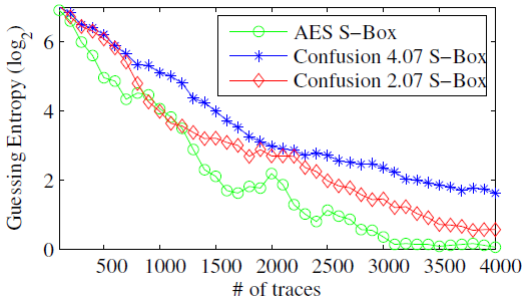


Figure 2.2: Constructing new AES-like Sboxes with improved confusion coefficient and comparing to the original Sbox.

common practice), he would observe increased side-channel resistance. Figure 2.2 aptly described the effect of such structures.

However, all such Sboxes had a very specific univariate adversary in mind. What was not considered instead was a multivariate advesary that could observe the (possibly profiled) leakage of key addition operations, internal Sbox computations and other bit/byte permutation operations that revealed useful information. Structures with enhanced confusion/transparency are in fact a prime target for horizontal attacks. Moving only slightly beyond the standard CPA scenario can potentially diminish the resistance gain by including more leaky observables and by using more informative leakage models. Sticking to the standard CPA scenario can lead to slightly misleading results about the security of a device.

2.4.2 Side-channel based intrusion detection systems

The second scenario under discussion condisers embedded devices, albeit in a non-cryptographic context. Analytically, we consider an embedded system that is part of any critical infrastructure. The system typically consists of an industrial control system that is in direct control of a process, such the water level control of a hydroelectric plant, the power grid distribution of an electric company or even the uranium enrichment process of a nuclear facility. Such embedded systems are good targets for any malware that aims to corrupt the industrial process and possibly cause peril.

The historic lack of security in these devices led to research towards unconventional protection mechanisms. The work of Van Aubel et al. [10] developed a side-channel-based intrusion detection system in a real world scenario, using measurements of the electromagnetic emissions from the processor on a Siemens Simatic S7-317 Programmable Logic Controller (Figure 2.3). The goal of the intrusion detection system was to observe the side-channel leakage during the regular modus operandi of the controller and profile such behavior. Subsequently, the current leakage of the device is compared to the regular profile and anomaly detection is performed.

The initial templating attempts relied largely on univariate techniques for profiling and comparison. Such techniques were adequate to detect large deviations from the correct device behavior. For instance, a malware that included extra operations in the

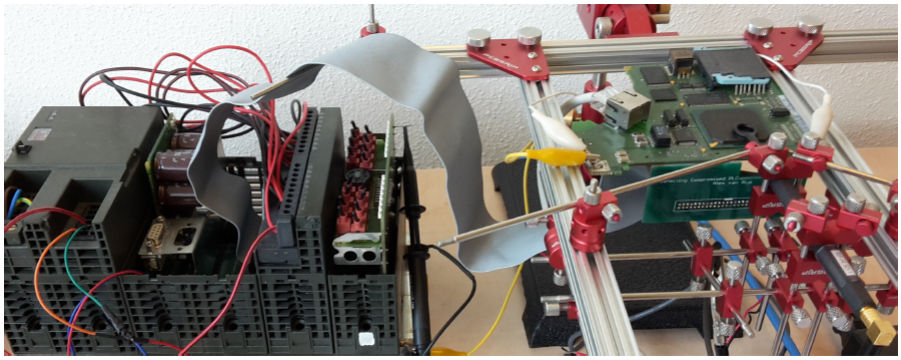


Figure 2.3: Side-channel intrusion detection system deployed on a Siemens programmable logic controller.

system was fairly easy to detect. However, if the malware inserted exhibited a behavior that was very close to the regular operation, detection became substantially harder. To improve, multivariate templating was deployed, improving the false-positive/negative rate, due to the fact that it was able to observe the full horizontal leakage of the regular operation.

2.4.3 Brain side-channels as biomarkers

The third and final scenario under discussion does not relate to embedded devices. Instead it focuses on the analysis of the biological-world counterpart of a CPU: the brain of a living organism. This particular scenario stems from the field of cognitive neuroscience, involving aging and sleep. Research in this field is particularly interested on how the brain activity, as observed via electroencephalography, conveys information about a living organism. In other words, the brain activity is considered a "biomarker" that captures the brain age and lets us succinctly draw conclusions about the health of an individual [158].

Common biology experiments involve analysis using standard Pearson correlation together with statistical significance intervals. In the case under discussion we have observed the side-channel information from the brain of 6 mice groups while they are sleeping. The mice groups include 3 aging levels namely young, middle-aged (18 months) and old (24 months). For every aging level, there exist mice that exercise regularly and mice that do not exercise. Applying standard correlation techniques on the brainwave activity resulted in a blurry picture (Figure 2.4). Most mice groups were close to each other (within statistical error margins), i.e. the electroencephalogram did not act a a strong biomarker. Once again, we applied multivariate statistical templates, aiming to extract more horizontal information from the brainwaves. The results are visible in Figure 2.5, where the distinction between different groups is substantially more visible, solidifying the brainwave as a relevant biomarker.

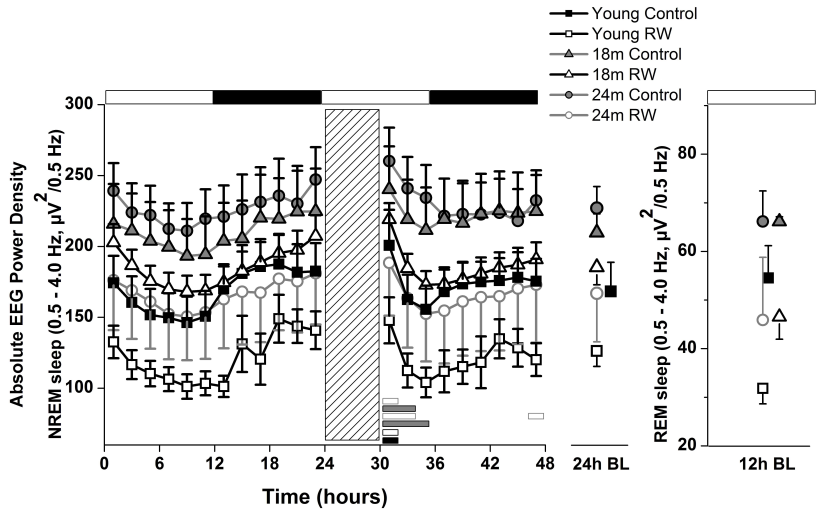


Figure 2.4: Correlation-based analysis of mice brainwaves.

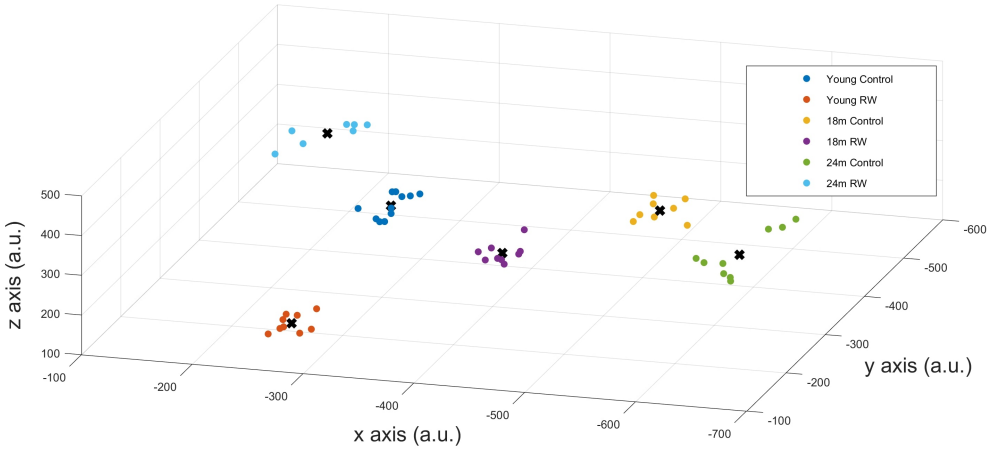


Figure 2.5: Template-based analysis of mice brainwaves.

Chapter 3

Masking Theory and Practice

"A theory is merely a scientific idea controlled by experiment."

Claude Bernard, 1865

To tackle various side-channel attacks, researchers have put forward a plethora of countermeasures, ranging from electrical layer mechanisms [167] to the algorithmic protection of a cipher implementation [49, 216]. After identifying a side-channel vulnerability, the common modus operandi of the community is a three-stage process of designing a suitable countermeasure against it, implementing it efficiently and finally performing an evaluation, using the appropriate laboratory apparatus and fulfilling Common Criteria guidelines[1]. The majority of available countermeasures is developed using this three-stage process, resulting in a conglomeration of 1) theoretical designs and formal security proofs, 2) high performance cryptographic implementations in assembly or HDL and 3) experimental evaluations using statistics and signal processing techniques.

Among various countermeasure options, masking [49] has drawn a large amount of attention. Originating from multi-party computation, masking can be applied on many layers, starting from the algorithmic layer of a cipher and reaching the gate layer. In all cases, it is capable of randomizing the intermediate values of its computation. Examining masking in the context of the afore-mentioned three-stage process, reveals the core research trends behind it. Namely, in the design stage, current research has provided masking with provable security features, guaranteeing through simulation proofs or formal verification that the countermeasure is secure against certain adversarial models [20, 111]. Continuing to the implementation stage, the large performance bottlenecks in masking schemes prompted high-speed or low-footprint implementations [92]. Finally, in the experimental evaluation stage, researchers try to address the gap between theoretical leakage models and actual device emissions [12], aiming to armor a provably secure masking scheme in the real world. The quote by Claude Bernard demonstrates exactly this ceaseless cycle between new theoretical models (along with corresponding secure schemes) and actual experiments that control, justify or disprove such theories.

[1] https://www.commoncriteriaportal.org/products/

This chapter is based on work published in [70, 94, 161] and is motivated by the need for fast and secure masking schemes for common microcontrollers. Thus, it provides masked cipher implementations of popular symmetric ciphers such as PRESENT and AES that break the current speed records, using carefully crafted assembly implementations. Moreover, this chapter demonstrates in several occasions how seemingly safe masking schemes can fall pray to hazardous electrical effects that reduce the schemes' security. The main points of the chapter are summarized below.

- This chapter follows the three-stage process of designing, implementing and evaluating a countermeasure in order to provide efficient and secure masking schemes. As a result, it shows that even higher-order masking schemes are within the reach of common ARM and AVR microcontrollers, should we choose to implement them carefully.

- This chapter addresses the significant gap between theory and practice in masking schemes. Using an experiment-driven approach, it identifies several electrical effects that damage the security of masking schemes and provides solutions that harden our implementations.

3.1 Introduction

Cipher implementors have observed that the masking countermeasure can imply a severe performance overhead in terms of processing speed due to the quadratic computational complexity required [111]. This is a reason why in practice, well-protected implementations are not as ubiquitous as one would hope. In software, higher-order masked implementations are typically orders of magnitude slower compared to unprotected implementations, as was demonstrated by Goudarzi et al. [92]. To reduce the performance overhead in ARM architectures, this chapter explores data parallelism both in its "artificial" version (register bitslicing), as well as in its real version (NEON parallel processing instructions). In both cases, a single instruction operates concurrently on multiple data elements inside one register, enabling faster computations on masking shares. Standard bitslicing [32,68] is used for the PRESENT cipher [36] on ARM Cortex-M4 and NEON instructions are used for the AES cipher [69] on ARM Cortex-A8.

In addition, it is well understood that masking can secure implementations against a *specific* threat model. Masking proofs are often conducted under the probing model [111], also encountered as the value-based leakage model in the literature. The underlying assumption of this model is that the adversary can only observe a single intermediate value with every probe used, and this assumption is often referred to as the *independent leakage assumption* [179]. Unfortunately, the exact computation of values in complex devices is not always visible at higher abstraction layers, e.g. the assembly code of a software implementation offers limited visibility on computations and thus cannot guarantee that the device adheres to this adversarial model. For instance, devices often exhibit distance-based leakages, which can reduce the security of the masking countermeasure [12]. Likewise, coupling effects [58], glitches [140] and other model divergences can pose similar security hazards.

3.1.1 Chapter Contribution

Starting, this chapter improves the current state of the art by creating an efficient, bit-sliced, 2nd-order masked implementation of PRESENT. The PRESENT cipher was selected due to its widespread applicability in the Internet of Things context. Our implementation requires 1644 bytes of RAM, 1552 bytes of Flash and encrypts 32 blocks of data in 209023 clock cycles, achieving a throughput of 6,532 clock cycles per block, excluding the cost of random number generation. Thus, the chapter demonstrates that midrange ARM-based architectures (Cortex-M) can host masked implementations efficiently, given that the implementors opt for full-scale assembly programs and use efficient bitsliced state representations.

In the same spirit, this chapter studies how the powerful NEON vector unit on high-end ARM Cortex-A8 processors can be used to obtain efficient masked AES implementations. We use again a bitsliced representation of the AES state and we exploit the data-level parallelism of Sbox computations. We provide the fastest publicly available higher-order masked AES implementations with 3rd & 8th-order security for the ARM Cortex-A8, requiring 7597 clock cycles per block, 11520 RAM bytes and

44004 Flash bytes for the 3rd-order version and 23616 clock cycles per block, 25600 RAM bytes and 70188 Flash bytes for the 7th-order version, excluding random number generation. Thus, we demonstrate that high-end ARM-based architectures with parallel processing capabilities can efficiently host highly protected implementations.

In addition to high-speed masked implementations, this chapter attempts to bridge the gap between theory and practice in masking countermeasures in the following manner. First, the chapter examines potential out-of-model leakages in ARM architectures, i.e. it performs in-depth side-channel experiments in order to test whether our masked ciphers in ARM Cortex-M4 and ARM Cortex-A8 are prone to order reduction. Continuing, it focuses on the ATMega163 microcontroller, taking advantage of its fairly noiseless leakage in order to investigate experimentally which effects violate ILA. Aided by this investigation, this chapter attempts to mitigate such problems and crafts the first (to our knowledge) 1st-order masked implementation in ATMega163 that is capable of resisting 1st-order, univariate attacks. In other words, it enforces the ILA in order to severely limit the informativeness of 1st-order leakages, forcing the adversary to resort to 2nd-order attacks. As a proof of concept, it develops a "hardened" 1st-order, ISW-based [111], bitsliced Sbox for the RECTANGLE cipher [224]. The "hardened" implementation requires 1319 clock cycles, a 15-fold increase compared to a "naive" 1st-order, ISW-based, bitsliced Sbox of the same cipher.

3.1.2 Previous Work

In this section, we initially describe the work of those practitioners that addressed the implementation of PRESENT and AES in software.

PRESENT implementations. Regarding protected implementations, Rauzy et al. presented a design methodology for inserting Dual-rail with Precharge Logic (DPL) in a software implementation of PRESENT in an automatic way [174]. They relied on an 8-bit AVR ATMega 163 implementation, bitsliced. They require 235,427 cycles for obtaining a single block of ciphertext. Regarding unprotected implementations, Poschmann implemented PRESENT in different software platforms [168]. In a 4-bit ATAM893-D at 2,000 KHz he obtained a performance of 55,734 cycles per block, on an 8-bit ATMega at 4 MHz, a performance of 10,089 cycles and on an 16-bit Infineon C167CR a performance of 19,460 cycles. Papagiannopoulos et al. [159] reached to 8,712 cycles per block by relying on a merged SP layer. Papagiannopoulos [162] also presented a bitsliced implementation of PRESENT on the 8-bit AVR ATiny85. He applied bitslicing to the permutation and substitution layers using a bitslice factor of 64. This improvement work relied on the PRESENT Sboxes generated after the application of 2-stage Boyar-Peralta heuristic in tandem with SAT solvers [40]. He obtained a throughput (cycles per block) of 2,967 using 3,816 bytes of Flash and 256 bytes of SRAM. In this chapter, we rely on the same Sbox. Dinu et al. also analyzed the suitability of a wide range of lightweight block ciphers in sensor-based applications in three different architectures: an 8-bit ATMega, 16-bit MSP430 and 32-bit ARM processor. They do not apply bit-slicing and implement the Cipher

Block Chaining (CBC) and counter mode (CTR) modes of operation [73]. The CBC implementation requires 121,906 cycles on the ATMega processor whereas the CTR implementation can obtain one block of ciphertext in 15,239 cycles. On the MSP430, the CBC and CTR modes of PRESENT, obtained a performance of 100,786 and 12,226 cycles respectively. The CBC implementation requires 138,947 cycles on the 32-bit ARM processor whereas the CTR implementation can obtain one block of ciphertext in 16,919 cycles. Last, the authors from [82] implement PRESENT-80 in an 8-bit ATtiny, requiring 11343 cycles, 1,000 bytes of code and 18 bytes of RAM.

AES implementations. Regarding protected implementations, Goudarzi et al. [92] compared the performance of different higher-order masking approaches on ARM architectures. A simplified model is assumed for the number of cycles that specific instructions take, without referring to a specific microarchitecture. Private communication made clear that they are derived from the Keil simulator based on an ARM7TDMI-S. Their fastest bitsliced implementation is claimed to take 120,972 cycles with 4 shares and 334,712 cycles with 8 shares. To achieve this performance, the presence of a fast TRNG is assumed that delivers fresh randomness at 2.5 cycles per byte. Only the cost of a normal `ldr` instruction it taken into account, which corresponds to our performance with pre-loaded randomness. Wang et al. [219] presented a masked AES implementation for NEON that appears to run in 14,855 cycles with 4 shares and 77,820 with 8 shares on a Cortex-A15 simulator. This uses a cheap LFSR-based PRNG to provide randomness. Balasch et al. [13] use the ARM Cortex-A8 for masked AES but they do not mention the performance of their implementation, since they focus solely on the security evaluation . Finally, Journault and Standaert [115] consider a bitsliced AES implementation with up to 32 shares on an ARM Cortex-M4. They exploit the parallelism of the shares, but not of AES itself as there are only 32-bit registers. An on-board TRNG is used to provide randomness at a reported speed of 20 cycles per byte. They use the refreshing and multiplication algorithms of [21] and report that 2,783,510 cycles are required to compute an AES block fwith 32 shares, of which 73% are spent on generating randomness.

Regarding unprotected implementations, bitsliced AES implementation of Bernstein et al. [30] uses exploits NEON to run at 19.12 cycles per byte (i.e., 306 cycles per block) in CTR mode, processing 8 blocks in parallel. Using similar approaches, the implementation of Käsper et al. [119] manages 7.59 cycles per byte on an Intel Core 2, while being constant-time.

3.1.3 Chapter Organization

This chapter is organized as follows. Section 3.2 describes a high-performance implementation of 2nd-order masked PRESENT on ARM Cortex-M4 and provides a side-channel evaluation that investigates order reduction. Similarly, Section 3.3 describes a high-performance implementation of 3rd & 7th-order masked AES and provides a side-channel evaluation that investigates order reduction, estimation/as-

sumption errors and upper bounds for attacks. Section 3.4 investigates order reduction and ILA-breaching effects, working towards mitigation on AVR ATMega163. Section 3.5 provides conclusions and future directions.

3.2 Masking PRESENT in ARM Cortex-M

The current section describes the design choices investigated in order to develop a protected, high-throughput, assembly-based PRESENT implementation. Sections 3.2.1 describes the logic-level and architectural optimizations performed, while Section 3.2.2 performs a side-channel evaluation.

3.2.1 Implementation of Higher-Order PRESENT

Inspecting the PRESENT cipher and its bit-based permutations layer, we note that CPU architectures tend to operate best on their native word size or half-words and they encounter performance issues with bit-level manipulation. Although the Cortex-M4 features bit-banding support[2], as well as a wide selection of bit-field instructions, applying them in the context of PRESENT requires extensive use of load and store instructions or numerous bit extractions/insertions, often resulting in poor performance. Therefore the natural implementation choice is opting for bitslicing, which was used to speed up DES bit-based permutations in a similar fashion [32].

In our implementation, we employ a bitsliced representation of factor 32, i.e. we process in parallel 32 cipher PRESENT-80 blocks, 64 bits each, resulting in 256 bytes per bitsliced encryption. Doing so, allows us to efficiently compute both the substitution and the permutation layer of the cipher. Analytically, the Sbox can be decomposed into $GF(2)$ operations which can be accelerated by via the SIMD-like instructions and it no longer requires the application of memory lookup tables.[3]. Similarly, the bit-based permutation layer can be accelerated by directly exchanging the memory contents of the corresponding bitsliced bits according to the permutation pattern, instead of relying on bit extraction, insertion and shifting.

3.2.1.1 ISW Multiplication & PRESENT Sbox

Efficient $GF(2)$ decomposition of the Sboxes has always sparked research in the direction of optimized of boolean circuits. In our implementation, we use the optimized boolean circuit suggested for PRESENT by Courtois et al. [64]. The optimized representation was generated by applying the Boyar-Peralta heuristic [40], which reduces the circuit's *gate complexity*, i.e. the number of AND, OR, XOR, NOT operations. The representation is shown below.

```
T1 = X2^X1;   T2 = X1&T1;   T3 = X0^T2;  Y4 = X3^T3;
```

[2]Bit-banding allows individual bits to be addressed as though they were bytes in RAM

[3]Note that implementations based on lookup tables can be prone to timing side-channel attacks in the presence of memory caches

```
T2 = T1&T3;     T1 ^= Y4;      T2 ^= X1;      T4 = X3|T2;
Y3 = T1^T4;  X3 =~ X3;         T2^ = X3;      Y1 = Y3^T2;
T2 |= T1;     Y2 = T3^T2;
```

Values X1–X4 represent an Sbox input, T1–T4 hold temporary values and Y1-Y4 are output values. The total cost is 14 operations, 4 non-linear (AND, OR) and 10 linear (XOR, NOT).

The bitsliced representation of PRESENT and the Sbox is decomposed into $GF(2)$ operations makes ISW [111] is our technique of choice in order to apply 2nd-order protection on the Boolean operations required for the Sbox computation. Table 3.1 shows the ISW equivalent of common Boolean operations when applied to bitsliced operands a and b, as well as the computational cost involved for each operation. The values $z_{i,j}$ where $1 \leq i < j \leq (d+1)$ are drawn from a uniform random distribution and the remaining $z_{i,j}$ are computed using $(z_{i,j} \oplus a_i b_j) \oplus a_j b_i$. Note that the cost of the NOT operation is a single negation, the cost of the XOR operation is linear and the cost of the AND,OR operations is quadratic. In our implementation, the OR operation is converted to a single AND and three NOT operations in order to apply the ISW method.

Table 3.1: ISW equivalents of common boolean operations

Operation	**ISW Equivalent**	**Cost**
NOT(a)	$\neg a_0$	$\mathcal{O}(1)$
XOR(a,b)	$a_i \oplus b_i$	$\mathcal{O}(d)$
OR(a,b)	**NOT(AND(NOT(a),NOT(b)))**	$\mathcal{O}(d^2)$
AND(a,b)	$a_i b_i \oplus \bigoplus_{i \neq j} z_{i,j}$	$\mathcal{O}(d^2)$

It comes as no surprise that the quadratic computational complexity of non-linear operations can result in a computationally demanding masked Sbox. To avoid this, several techniques [46, 47, 64, 91, 202] work towards reducing the *multiplicative complexity of an Sbox*, i.e the number of AND,OR operations. The $GF(2)$ decomposition that we currently use is optimal w.r.t. multiplicative complexity, since brute-force techniques [95] demonstrate that the minimal complexity in $GF(2)$ of cryptographically relevant, 4-bit Sboxes is 4 non-linear operations.

3.2.1.2 ARM-based Optimizations

Our implementation targets the ARM Cortex-M4 microcontroller architecture using ARM assembly with Thumb2 encoding. Thus, we use a 32-bit architecture with 14 general purpose registers designed for low-cost, low-power applications. The implementation board is the Riscure Pinata which is based on the STM32F417IG SoC by ST and embeds an ARM 32-bit Cortex-M4 CPU clocked at 168 MHz. It features 1,024 Kbytes of Flash and 196 Kbytes of RAM. The device is also equipped with a TRNG on the board in order to generate the random values associated to our

masking implementation. In the case of the STM32F417IG, the TRNG generates 32-bit random numbers via an integrated analog circuit. Note that the computational penalty w.r.t. random number generation is particularly steep when implemented on-the-fly, mounting to roughly 25% of the total computation. Still, we note that the random numbers can be precomputed in advance, given that the application context allows for time intervals between consecutive encryptions. Below, we discuss implementation details and efficiency improvements pertaining to the ARM architecture, memory organization and assembly instructions.

- **Memory organization**: Our design requires two full bitsliced states in RAM, each comprising of three sub-states corresponding to the three-share masking scheme. The two full bitsliced states are needed because the permutation layer would otherwise overwrite unprocessed data. We optimize for cycles by integrating the permutation into the Sbox and writing words to their permuted destination immediately after the Sbox computation.

  Wherever the code operates on shares we organize our fetch and store data in batches so as to reduce overhead. In most cases we use the LDM and STM instructions to load or store three or four words at a time. This yields improvements in the Sbox computation when reading in the next four words to be substituted, in the key schedule, where three words at a time are read in for processing and also when converting a regular state representation from/to a bitsliced one.

- **Loop Unrolling**: To improve the efficiency of our Sbox implementation, which encrypts twelve shares (four bit-sliced data blocks of three shares each), we unroll the substitution process to reduce the unnecessary read/write steps required for a looped construction. The unrolling adds considerable size to the code, yet we achieve trading code size for throughput. Note that unrolling is performed with memory access in mind. For example, we mentioned that adding the key schedule is performed in a loop of three words. This optimizes the key schedule operation and maximizes the amount of data we can bring from/to the RAM.

- **Key Schedule**: The round key is not stored in a bitsliced fashion and the key schedule is computed on the fly. Note that round key precomputation is also a valid implementation option, assuming that the key does not need to be renewed often. Since, key refreshing can act as a side-channel countermeasure, we chose to retain the on-the-fly key updates. Updating the round key requires a push through the Sbox for four bits each round. To that purpose, we use Cortex-M4's UBFX instruction for extracting a contiguous series of bits from a word in an efficient manner. In addition, we used ARM's barrel shifter function, which allows the second operand to be shifted with no additional cost before an instruction is performed.

3.2.1.3 Performance Results

The current section summarizes the achieved performance results with respect to throughput and size. We depict in Tables 3.2 and 3.3 our performance figures, compared to previous works. We outperform prior art on the same architecture between 2.5 and 21.2 times. As expected, the ISW implementation of the Sbox dominated CPU time, accounting for 95,88% of all clock cycles within the encryption process. A complete breakdown of the memory and time overheads required for different modules is provided in Table 3.4.

Table 3.2: PRESENT implementations, comparison with prior art (performance)

Work	Implementation	Bitslicing	Bitslicing factor	Protected	Platform	No. cycles per block
This work	PRESENT-80	**yes**	**32**	**yes**	**ARM Cortex–M4**	**6,532**
[73]	PRESENT-80, CBC	no	-	no	ATMega	121,906
[73]	PRESENT-80, CBC	no	-	no	MSP430	100,786
[73]	PRESENT-80, CBC	no	-	no	ARM	138,947
[73]	PRESENT-80, CTR	no	-	no	ATMega	15,239
[73]	PRESENT-80, CTR	no	-	no	MSP430	12,226
[73]	PRESENT-80, CTR	no	-	no	ARM	16,919
[159]	PRESENT-80	**yes**	8	no	ATiny	8,721
[174]	PRESENT-80	**yes**	8	no	ATMega163	78,403
[174]	PRESENT-80, DPL	**yes**	8	**yes**	ATMega163	235,427
[162]	PRESENT-80	**yes**	8	no	ATiny85	2,967
[168]	PRESENT-80	no	-	no	ATAM893-D	55,734
[168]	PRESENT-80	no	-	no	ATMega163	10,089
[168]	PRESENT-80	no	-	no	C167CR	19,460

Table 3.3: PRESENT implementations, comparison with prior art (size)

Work	Implementation	Code (bytes)	RAM (bytes)
This work	PRESENT-80	1,548	1,644
[162]	PRESENT-80	3,816	256
[168]	PRESENT-80, ATMega	1,494	272
[168]	PRESENT-80, C167CR	$45.9 \cdot 10^3$	-
[73]	PRESENT-80, CBC, ATMega	1,388	56
[73]	PRESENT-80, CBC, MSP430	1,108	52
[73]	PRESENT-80, CBC, ARM	1,304	124
[73]	PRESENT-80, CTR, ATMega	1,416	54
[73]	PRESENT-80, CTR, MSP430	1,244	58
[73]	PRESENT-80, CTR, ARM	1,532	140
[159]	PRESENT-80	1,794	-
[174]	PRESENT-80, bitslicing	1,620	288
[174]	PRESENT-80, bitslicing + DPL	3,056	352

3.2.2 Side-Channel Evaluation of Higher-Order PRESENT

In this section, we assess experimentally the security level (masking order) provided by the ISW masking scheme, taking into account the possibility of distance-based leakages in ARM Cortext-M4. In addition, we investigate whether the theoretical repercussions of distance-based leakages can be confirmed experimentally. In other words, we examine whether the cost of "lazy engineering" as introduced by Balasch et

Table 3.4: SW transformations of common logical operations

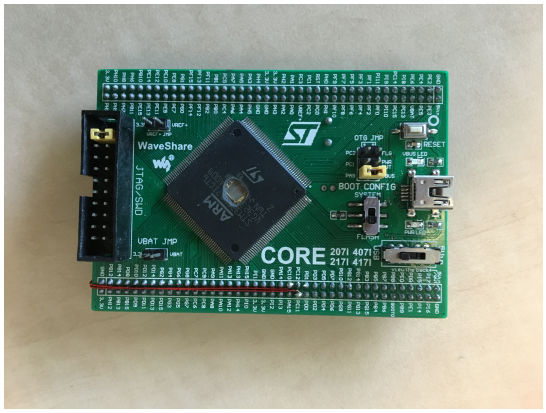
Operation	Code Size (%)	No. Cycles (%)
main	208 (13.44)	3,807 (1.82)
Sbox	892 (57.62)	200,404 (95.88)
updatekey	146 (9.43)	1,688 (0.81)
addroundkey	176 (11.37)	1,209 (0.58)
split data	60 (3.88)	1,292 (0.62)
unsplit data	66 (4.26)	623 (0.30)

al. [12] is applicable to an ARM-based microcontroller. Section 3.2.2.1 describes the experimental setup. Section 3.2.2.2 evaluates the security order of the implementation.

3.2.2.1 Experimental Setup

The acquisition is performed on the ARM-based Pinata device [4], using a Picoscope 5203 oscilloscope, the Riscure current probe [5] and the Riscure Inspector toolchain. The device clock operates on 168 MHz and the oscilloscope's sample rate is 1 GSample/sec. The PRESENT process sets a GPIO port high before the execution of PRESENT and sets it low after PRESENT is finished, so that it can be used as the trigger signal. We also apply post-processing in the form of simple static alignment signal resampling at the clock frequency. The Pinata device is visible in Figure 3.1.

Figure 3.1: Modified Pinata ARM STM32F417IG device.



3.2.2.2 Security Order Evaluation

The *effective* and *efficient* evaluation of the *actual mask order* of cryptographic implementations remains an open problem due to several evaluation pitfalls. Effectivity-wise, when evaluating a masking scheme via the measured power consumption, we face the pitfall of the *limited attack scope*. That is, a particular attack technique in use may

[4] https://www.riscure.com/security-tools/hardware/pinata
[5] https://www.riscure.com/security-tools/hardware/current-probe

fail to exploit the available leakage due to e.g. an unsuitable choice of intermediate values or an incorrect power model assumption[6]. Moreover, introducing additional countermeasures on top of the masking scheme may render particular exploitation techniques ineffective, while the implementation remains vulnerable to different lines of attack.

In order to tackle this issue, the community followed several approaches. Prior research established generic side-channel distinguishers such as Mutual Information Analysis (MIA) [23], the Kolmogorov-Smirnov and the Cràmer-von Mises tests [217, 221], which require minimal assumptions about the noise and the power model of the device under test. On the other side of the spectrum, Standaert et al. [197] proposed an evaluation framework assuming the strongest possible adversary, equipped with extensive profiling capabilities and Bayesian templates. While being effective, the afore mentioned approaches focus on leakage *exploitation* and perform key recovery, which may require a large number of traces. Thus, they face the *efficiency* pitfall w.r.t. computational and storage requirements. Note that this increased demand for resources is magnified when inserting extra countermeasures in a masked implementation. Thus, it can be difficult to decide with confidence whether the masking order is reduced or not.

In order to evaluate the *effective masking order* [134], we opt for the more recent approach called leakage detection methodology [60]. This approach focuses on leakage detection and disregards exploitation. Thus, the acquisition and the computational cost is reduced while the methodology can retain its generic nature. Despite the gain achieved via decoupling detection and exploitation, the leakage detection methodology still presents challenges w.r.t. efficiency. In the context of software masking, we need to combine multiple time samples in order to evaluate the masked implementation. Thus, we rely on the work by Schneider et al. [185], who extended the leakage detection methodology into higher-order evaluations by providing efficient, incremental formulas that can handle the computation involved with minimal memory requirements. In certain cases, we also resort to traditional evaluation techniques such as correlation-power analysis (CPA) [41], despite their limited attack scope, so as to enhance our discussion.

In order to perform leakage detection and determine the actual masking order, we use the fixed vs. random, non-specific t-test statistic. The process involves two steps: a custom acquisition of two trace sets (populations) and a population comparison based on statistical inference. In the first step, we perform a fixed vs. random acquisition and obtain two distinct trace sets for comparison: S_{fixed} and S_{random}, under the same encryption key. For S_{fixed}, the input plaintext is set to a fixed value, while for S_{random}, the input is drawn from a uniformly random distribution. Following the suggestion from Shneider et al. [185], the implementation receives the fixed or random plaintext in a non-deterministic and randomly-interleaved manner. This type of acquisition is performed in order to randomize the implementation's internal state and avoid measurement-related variations over time, e.g. due to environmental parameters. The evaluation test to be performed is non-specific, i.e. we target

[6]Knowledge about the device can often be limited in the context of black-box evaluations.

all sensitive values computed during encryption. Thus, we maintain a wide attack scope, without any prior assumptions on the leakage model or intermediate values. For the second step, we model the sets S_{fixed} and S_{random} as independent random samples $\{S^1_{fixed} \dots S^n_{fixed}\}$ and $\{S^1_{random} \dots S^m_{random}\}$ drawn from normal distributions with means $\mu_{fixed}, \mu_{random}$, standard deviations $\sigma_{fixed}, \sigma_{random}$ and $\sigma_{fixed} \neq \sigma_{random}$. Subsequently, leakage detection methods will test the equality of means μ_{fixed}, μ_{random} (null hypothesis). Finding a statistic for this test is known as the Behrens-Fisher problem and an approximate solution is the Welch t-test [130] with υ degrees of freedom, as shown below.

$$\begin{aligned} H_{null}: &\quad \mu_{fixed} = \mu_{random} \\ H_{alt}: &\quad \mu_{fixed} \neq \mu_{random} \end{aligned} \tag{3.1}$$

$$w = \frac{\mu_{fixed} - \mu_{random}}{\sqrt{\frac{\sigma^2_{fixed}}{n} + \frac{\sigma^2_{random}}{m}}} \tag{3.2}$$

$$\upsilon = \frac{(\frac{\sigma^2_{fixed}}{n} + \frac{\sigma^2_{random}}{m})^2}{\frac{\sigma^4_{fixed}}{n^2(n-1)} + \frac{\sigma^4_{random}}{m^2(m-1)}} \tag{3.3}$$

The null hypothesis H_{null} is rejected at a given level of significance α, if $|w| > t_{\alpha/2,\upsilon}$, where $t_{\alpha/2,\upsilon}$ is the value of the Student t distribution with υ degrees of freedom[7]. In the evaluation context, rejecting H_{null} implies leakage detection, i.e. potential evidence of an ineffective masking scheme. A common rejection criterion that we also use in our analysis is $|w| > 4.5$, which corresponds to $\upsilon > 1000$ and $\alpha > 0.99999$ [72]. Note that that H_{null} rejection shouldn't be interpreted directly as an applicable vulnerability. Even after detection, the amount of traces required for exploitation may render an attack infeasible.

In this work, we need to evaluate the masking order provided by our ARM-based, 2nd-order masked cipher. From a theoretical point of view, a 2nd-order ISW masking countermeasure is capable of preventing value-based leakages of order 2 or less. However, practice has demonstrated that software implementations, including ARM microcontrollers, may exhibit leakages with large divergence from the value-based leakage abstraction. An exemplary case is the distance-based leakage model, observed by Daemen et al. [208], addressed by Coron et al. [63] and recently formalized by Balasch et al. [185]. This particular divergence leads in the reduction of the security order. Repeating Balasch et al., a dth-order scheme can reduce to order $\lfloor \frac{d}{2} \rfloor$ and to identify such order reduction in our implementation, we use the Welch t-test in order to verify experimentally the theoretical security claims.

We commence the evaluation by testing the 1st-order security of our masked cipher. We perform the 1st-order t-test on the first round of bitsliced PRESENT. The size of both S_{fixed} and S_{random} is 10k traces with 30k samples per trace. The t-test results are visible in Figure 3.2. We observe that that we remain well below the 4.5 threshold, indicating that our 2nd-order masked PRESENT implementation is able to maintain 1st-order security.

[7]Note that side-channel analysis usually employs two-tailed tests.

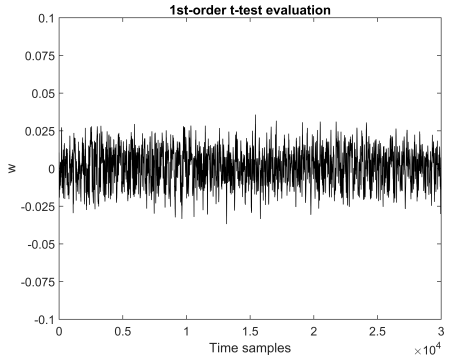


Figure 3.2: 1st-order t-test evaluation for 2nd-order masked PRESENT cipher. The results suggest absence of 1st-order leakage.

To enhance our confidence, we also perform a 1st-order CPA attack, with a large amount of traces (800k) to exploit potential 1st-order leakages. We use the HW model and a custom-made selection fuction due to the bitsliced Sbox computation. Similarly to Balasch et al. [13], the selection function must take into account that not all Sbox output bits leak at the same time due to the $GF(2)$-oriented Sbox implementation. Thus, our selection function focuses on key bits from different registers that once combined through the Sbox, affect a single bit of the Sbox output. Attacking a section of the 1st round with 10k traces, while the RNG is disabled, is successful, confirming the validity of our choice w.r.t. the leakage model (HW) and selection function. The results are visible in Figure 3.3. We also perform the CPA attack with enabled RNG and the results are visible in Figure 3.4. In order to manage the computation required, we employ the techniques suggested by Bottinelli et al. [39], i.e. we partition the 800k traces, compute correlation coefficient per partition, then recombine in order to reduce the execution and memory workload.

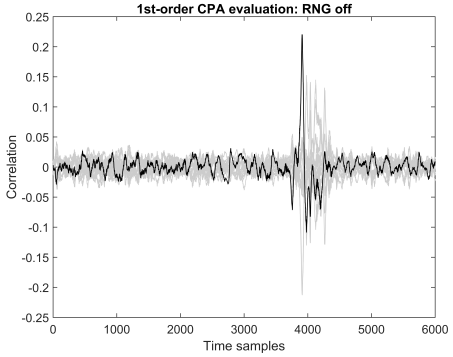


Figure 3.3: 1st-order CPA attack results with RNG turned off, using 10k traces in selected section of the 1st round.

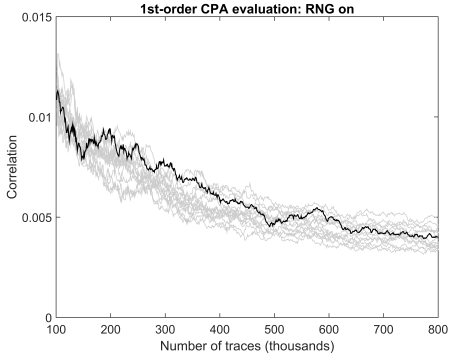


Figure 3.4: 1st-order CPA attack results with RNG turned on, ranging from 100k until 800k traces.

The results demonstrate that no 1st-order leakage can be exploited in the presence of our 2nd-order scheme. Both the t-test and the CPA result is in accordance with the order-reduction theorem, since a 2nd-order masked implementation can maintain

$\lfloor \frac{2}{2} \rfloor = 1$ order of security in the presence of distance-based leakages.

Assuming that our device exhibits distance-based leakage, it is of particular interest to prove experimentally that the order-reduction theorem holds when we test the 2nd-order security of our ARM-based masked implementation. Performing a 2nd-order evaluation requires pre-processing the acquired trace sets in order to generate all possible 2-tuples (pairs) of distinct samples via a combination function. Subsequently, the multivariate 2nd-order t-test is performed on the generated trace sets in order to determine the robustness of the 2nd order.

The main hindrance of this process is the computational complexity pertaining to generating and processing all $\binom{NoSamples}{2}$ sample pairs. Even with a small number of samples per trace, the evaluation cost can quickly become prohibitive. To address this issue, researchers have relied on intuitive selection of points of interest in conjunction with naive search [155] or they deployed heuristic techniques such as projection pursuits [80] to perform point of interest selection for higher-order attacks. In our evaluation, we follow the intuitive approach by focusing on a reduced version of the 1st round which contains the substitution layer. Inside this reduced round, we enumerate naively all possible pairs. Given the bitsliced nature of the implementation and the considerable RNG overhead, the reduced round has a length of 800 samples. In order to keep the processing cost manageable, we use again the incremental formulas suggested by Schneider et al. [185] which enable the efficient computation of the multivariate statistical moments required for 2nd-order t-tests. The memory-less feature of the computation yields significant improvement compared to straightforward computation techniques. In addition, we partition the reduced round into windows of 150 samples each and perform the attack in each window independently. Figure 3.5 shows the t-test results using 10k fixed input traces and 10k random input traces for the sample window with the largest detected leakage.

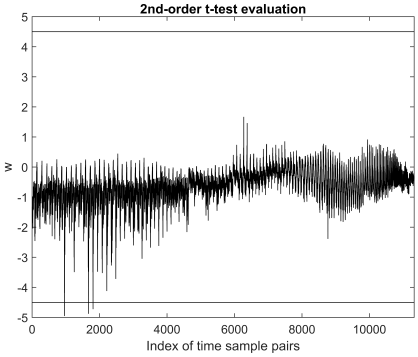


Figure 3.5: 2nd-order t-test results. The rejection of H_{null} indicates potential leakage.

The test value slightly exceeds the threshold, indicating potential leakage. Thus, it hints the experimental verification of the order-detection theorem in our ARM-based device for 2nd-order ISW schemes. However, several concerns were raised over the t-test robustness, usually regarding the exact threshold value ([12] – Appendix A, [72]). As a result, it remains an open question whether 2nd-order leakages are practically exploitable in our context. To investigate this, we perform a 2nd-order

CPA-based attack using the centered product combination function and the custom bitsliced selection function on the 1st round of PRESENT. The point selection window has size 100 samples and we use 100k traces. The results are visible in Figure 3.6 and show that the leakage is *exploitable* with roughly 60k traces.

As a result, we suggest that the order-reduction theorem remains applicable in software-based, masked implementations for the ARM Cortex-M4. However, we recommend that the exploitation is always verified in practice.

Moreover, we need to stress the fact that this type of behavior has been observed in a *specific* ARM-based device. Although it provides indications on the behavior of similar architectures, this experimental result should not be extrapolated as a hard fact w.r.t. all ARM Cortex-M devices. Naturally, a 3rd-order multivariate t-test is able to detect a large amount of leakage, as shown in Figure 3.7 and indicates that a 3rd-order attack is also applicable.

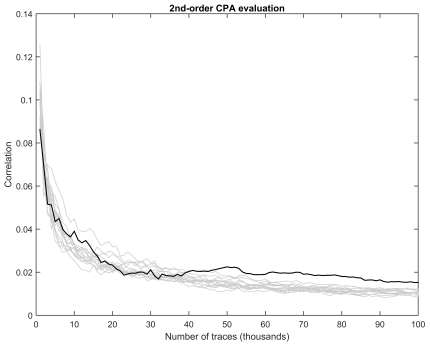


Figure 3.6: 2nd-order CPA results indicating exploitable leakage.

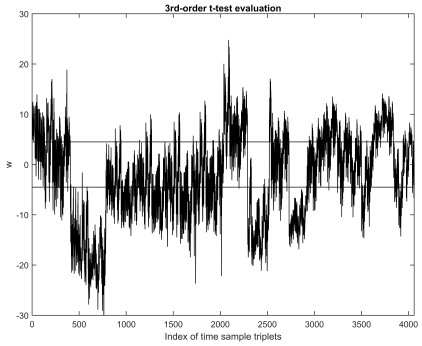


Figure 3.7: 3rd-order t-test results on a section of the 1st round, indicating strong 3rd-order leakage.

3.3 Masking AES in ARM Cortex-A

The current section describes the design choices investigated in order to develop a protected, high-throughput, assembly-based AES implementation. Section 3.3.1 describes the logic-level and architectural optimization performed, while Section 3.3.2 performs a side-channel evaluation.

3.3.1 Implementation of Higher-Order AES

The AES state [69] is usually pictured as a square matrix of 4 by 4 byte elements. This representation leads to efficient software implementations when SubBytes is implemented, aided by lookup tables. However, such implementations are also prone to cache-timing attacks [29], as the memory location of the value that is looked up depends on some secret intermediate value. An alternative bitsliced representation

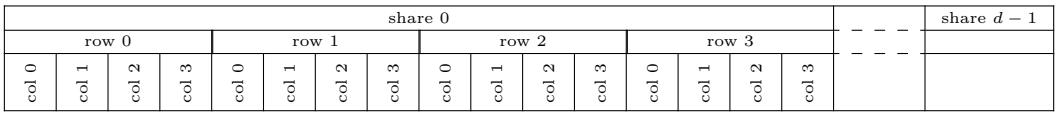


Figure 3.8: Register lay-out for the single-block implementations. There are 8 of these $16d$-bit vector registers. The cells on the bottom row represent individual bits.

avoids these attacks. In this bitsliced representation, all the first bits of every byte are put in one register, all the second bits in the next register, etc. For SubBytes, one can now compute the Sbox on the individual bits and do that for all 16 bytes in parallel. The Sbox parallelism of AES for bitslicing was first exploited by Könighofer in [127] and it was also used in the speed-record-setting AES implementation targeting Intel Core 2 processors by Käsper et al. [119]. At a small cost, the other (linear) operations of AES are modified to operate on this bitsliced representation as well.

However, on most devices registers are longer than 16 bits, so it would be a waste to not utilize this. AES implementations without side-channel protections choose to process multiple blocks in parallel, by simply concatenating multiple 16-bit chunks from independent blocks in one register. For example, the AES implementation of Käsper et al. [119] processes 8 blocks in parallel in a 128-bit XMM register. When the vector registers become larger, this trivially leads to higher throughputs for parallel modes of operation.

Similarly to bitsliced masked PRESENT, this implementation section we consider three implementations that, instead of multiple blocks, process multiple shares in parallel. The first implementation fills a 64-bit `d` register with 4 shares. The second has 8 shares, that are used to fill a 128-bit `q` register. The third combines 2 blocks with each 4 shares, and also utilizes the 128-bit `q` registers. It interleaves the shares of the 2 blocks for efficiency reasons. Note that this third implementation requires a parallel mode of operation. Continuing with the implementation, Section 3.3.1.1 discusses in detail the masked multiplication and refreshing, while Sections 3.3.1.2 and 3.3.1.3 focus on the non-linear and linear layers respectively. Performance figures are provided in Section 3.3.1.4.

3.3.1.1 Parallel Multiplication & Refreshing

Instead of using standard ISW-based masking, the AES implementations opt for the more recent methods of Barthe et al. [21], where new algorithms for parallel multiplication (including the `AND` operation) and parallel refreshing are proposed. They are proven to be secure in the bounded moment model and proven to be strongly non-interfering using techniques from automated program verification [20]. Correct implementations of these algorithms are critical for the security of our implementations. We consider slightly improved algorithms for 4 and 8 shares that require less randomness, but we could not generalize them to an improvement for all orders. As with the original algorithms, they are proven secure using the same automatic verification tools.

Refreshing. Refreshing can be necessary to make sure that values in registers are again statistically independent. The refreshing algorithm of Barthe et al. [21] requires $2d$ bytes of fresh uniform randomness. Let $\mathbf{v}_x$ denote a vector register that contains shares $(x_0, \ldots, x_{d-1})$, where $\bigoplus_{i=0}^{d-1} x_i = x$, and let $\mathbf{v}_r$ be a vector of the same length that contains uniformly random values. In the case of AES, a single share would be 16 bits long, so a randomness vector $\mathbf{v}_r$ will be $2d$ bytes. Then $\mathbf{v}_{x'} = \mathbf{v}_r \oplus \mathrm{rot}(\mathbf{v}_r, 1) \oplus \mathbf{v}_x$ is a secure way to refresh x, where $\mathrm{rot}(\mathbf{v}, n)$ rotates $\mathbf{v}$ to either left or right by n shares. Note that in the case of AES, this is equal to applying a rotation by $2n$ bytes.

For 4 shares, this algorithm additionally achieves SNI. However, to reach this with 8 shares, in [21] it turned out to be necessary to iterate the refreshing algorithm 3 times. In other words, one would need to compute

$$\mathbf{v}_r \oplus \mathrm{rot}(\mathbf{v}_r, 1) \oplus \mathbf{v}_{r'} \oplus \mathrm{rot}(\mathbf{v}_{r'}, 1) \oplus \mathbf{v}_{r''} \oplus \mathrm{rot}(\mathbf{v}_{r''}, 1) \oplus \mathbf{v}_x$$

to achieve SNI at order 7. This requires 3 vectors of uniform randomness, or 48 bytes with AES. We improve this algorithm by computing:

$$\mathbf{v}_r \oplus \mathrm{rot}(\mathbf{v}_r, 1) \oplus \mathbf{v}_{r'} \oplus \mathrm{rot}(\mathbf{v}_{r'}, 2) \oplus \mathbf{v}_x \,.$$

We verified with the current version of the tool of [20] that this also achieves SNI at order 7. Moreover, it requires one less randomness vector. In the case of AES, we now require 32 bytes of uniform randomness.

Multiplication. Multiplication in a finite field, or an `AND` gate in the case of $\mathbb{F}_2$, is trickier to perform in a secure way. Consider the case where one wants to compute $z = x \cdot y$. Let $\mathbf{v}_r$ and $\mathbf{v}_{r'}$ be uniformly random vectors. Then, with 4 shares, the algorithm suggested in [21] computes the following to achieve SNI at order 3:

$$\begin{aligned}\mathbf{v}_z = {} & \mathbf{v}_x \cdot \mathbf{v}_y \oplus \mathbf{v}_r \oplus \mathbf{v}_x \cdot \mathrm{rot}(\mathbf{v}_y, 1) \oplus \mathrm{rot}(\mathbf{v}_x, 1) \cdot \mathbf{v}_y \oplus \mathrm{rot}(\mathbf{v}_r, 1) \\ & \oplus \mathbf{v}_x \cdot \mathrm{rot}(\mathbf{v}_y, 2) \oplus \mathbf{v}_{r'} \oplus \mathrm{rot}(\mathbf{v}_{r'}, 1) \,.\end{aligned}$$

However, we can again improve this slightly such that less randomness will be necessary. Let r_4 be a uniformly random value. Then we proved using the tool of [20] that the following is also 3rd-order SNI-secure. For AES, this requires 10 fresh uniformly random bytes (8 for $\vec{r}$ and 2 for r_4) instead of 16:

$$\begin{aligned}\mathbf{v}_z = {} & \mathbf{v}_x \cdot \mathbf{v}_y \oplus \mathbf{v}_r \oplus \mathbf{v}_x \cdot \mathrm{rot}(\mathbf{v}_y, 1) \oplus \mathrm{rot}(\mathbf{v}_x, 1) \cdot \mathbf{v}_y \oplus \mathrm{rot}(\mathbf{v}_r, 1) \\ & \oplus \mathbf{v}_x \cdot \mathrm{rot}(\mathbf{v}_y, 2) \oplus [r_4, r_4, r_4, r_4] \,.\end{aligned}$$

With 8 shares, we use the original algorithm of [21] that is SNI at order 7. This requires 3 randomness vectors, which in the case of AES amounts to 48 bytes:

$$\begin{aligned}\mathbf{v}_z = {} & \mathbf{v}_x \cdot \mathbf{v}_y \oplus \mathbf{v}_r \oplus \mathbf{v}_x \cdot \mathrm{rot}(\mathbf{v}_y, 1) \oplus \mathrm{rot}(\mathbf{v}_x, 1) \cdot \mathbf{v}_y \oplus \mathrm{rot}(\mathbf{v}_r, 1) \\ & \oplus \mathbf{v}_x \cdot \mathrm{rot}(\mathbf{v}_y, 2) \oplus \mathrm{rot}(\mathbf{v}_x, 2) \cdot \mathbf{v}_y \oplus \mathbf{v}_{r'} \\ & \oplus \mathbf{v}_x \cdot \mathrm{rot}(\mathbf{v}_y, 3) \oplus \mathrm{rot}(\mathbf{v}_x, 3) \cdot \mathbf{v}_y \oplus \mathrm{rot}(\mathbf{v}_{r'}, 1) \\ & \oplus \mathbf{v}_x \cdot \mathrm{rot}(\mathbf{v}_y, 4) \oplus \mathbf{v}_{r''} \oplus \mathrm{rot}(\mathbf{v}_{r''}, 1)\end{aligned}$$

We attempted to reduce this by replacing the last randomness vector by a vector with a single random value, as in the algorithm for 4 shares, but we found that this does not achieve SNI at order 7.

Randomness. As we observed for masked PRESENT, implementations that are protected using higher-order masking require a lot of RNG and the randomness improvements in the $GF(2)$ multiplication work towards reducing it. Regardless of the amount of RNG, to be able to prove statistical independence, this randomness should be fresh and uniformly distributed. For resisting attacks in practice, it is not so clear whether the exact requirements are this strict. For instance, it might also be fine to expand a random seed using a pseudo-random number generator, or even to re-use randomness, as we will examine in Chapter 4. Since, the impact on the performance can be very significant, we consider various approaches that occur in the literature. The first is to read all the randomness that we require from `/dev/urandom` using `fread`, like Balasch et al. [13]. This is the most conservative approach, but it is rather slow. Second, we also consider the case where all required randomness is already in a file that needs to be read into memory, i.e. we assume that RNG is carried out during idle device phases before/after encryption. The third approach assumes that there exists a fast true random-number generator and only considers the cost of a normal load instruction (`vld1`), like in [92].

The AES implementation with 4 shares requires 8 bytes per refresh and 10 bytes per masked `AND`. In the next section we will see that this amounts to $10 \cdot 32 \cdot (8+10) = 5760$ random bytes in total for the full AES, excluding the randomness used to do the initial masking of the input and the round keys. Naturally, the implementation that computes two blocks in parallel requires double the amount of random bytes. For 8 shares, refreshing takes 32 bytes and a masked `AND` uses 48 bytes, which makes the total $10 \cdot 32 \cdot (32 + 48) = 25600$ bytes.

3.3.1.2 AES Sbox

Using the masked `AND` and refreshing algorithms, we can build our bitsliced Sbox. Several papers have presented optimized bitsliced representations of the AES Sbox. The smallest known to us is by Boyar and Peralta [40]. It uses 83 `XOR`s/`XNOR`s and 32 `AND`s, which was later improved to 81 `XOR`s/`XNOR`s and 32 `AND`s. The few `NOT`s can be moved into the key expansion, so we only need to consider `XOR`s and `AND`s. We use this implementation as our starting point, as this is also the implementation with the smallest number of binary `AND`s, and an `AND` will be much slower to compute than a `XOR`.

We use the compiler provided in [19] to generate a first masked implementation of SubBytes. This tells us when it is necessary to refresh a value, making sure that we do not refresh more often than strictly necessary. For our version of SubBytes, however, the compiler adds a refresh on one of the inputs for every `AND`. Then we implement an `XOR` on multiple shares in parallel with a `veor` instruction. For an `AND`, we use the algorithms of the previous section. Finally, the code has been manually optimized

to limit pipeline stalls.

The Sbox implementation has many intermediate variables. With 4 shares and a single block, the `d` registers are used. There are 32 of them and this turns out to be sufficient to store all the intermediate values. With two blocks or with 8 shares, however, we can use only 16 `q` registers. This implies that values have to be spilled to the stack. Of course, we want to minimize the overhead caused by this. In the work by Schwabe et al. [187], an instruction scheduler and register allocator for the ARM Cortex-M4 was used to optimize the number of pushes to the stack. We modify this tool to handle the NEON instructions that we need, and use it to obtain an implementation with 18 push instructions and 18 loads.

According to a cycle-count simulator [192], our SubBytes implementation takes 1035 cycles with one block and 4 shares and 2127 cycles with 8 shares.

3.3.1.3 AES Linear Layer

We now discuss the linear operations of AES. We manually optimized them using a cycle-count simulator to hide as many latencies as possible [192].

AddRoundKey. AddRoundKey loads the round key with the `vld1` instruction and adds it to the state using `veor`. The loads and arithmetic instructions can be interleaved. This helps because they go into separate NEON pipelines. An arithmetic instruction can than be executed in parallel with the load of the next part of the round key. For the loads, we make sure that they are aligned to at least 64 bits. AddRoundKey then only takes 10 cycles.

ShiftRows. With ShiftRows, rotations by fixed distances over 16 bits need to be computed. This can be implemented using `vand`, `vsra`, `vshl`, and `vorr` instructions. The arithmetic pipeline is now clearly the bottleneck. According to the simulator, our ShiftRows takes 150 cycles.

MixColumns. MixColumns requires more rotations by 4 or by 12 over 16 bits. This takes 106 cycles as measured by the simulator.

3.3.1.4 Performance Results

We benchmark our implementations on the BeagleBone Black with the clock frequency fixed at 1 GHz. In other words, we disabled frequency scaling. For the rest, we did not apply any changes to a standard Debian Linux 9 installation. In particular, we did not disable background processes and did not give our process special priority or CPU core affinity. The implementations are run 10000 times and the median cycle counts are given in Table 3.5.

When using `/dev/urandom`, more than 99% of the time is spent on generating randomness, which is delivered at a rate of only 369 cycles per byte in the 8-share case. With a faster RNG, it becomes clear that our implementations are very fast and practical. We reach 474 cycles/byte with 4 shares and 1476 cycles/byte with 8

	4 shares 1 block	4 shares 2 blocks	8 shares 1 block
Clock cycles (randomness from `/dev/urandom`)	1,598,133	4,738,024	9,470,743
Clock cycles (randomness from normal file)	14,488	17,586	26,601
Clock cycles (pre-loaded randomness)	12,385	15,194	23,616
Random bytes	5,760	11,520	25,600
Stack usage in bytes	12	300	300
Code size in bytes	39,748	44,004	70,188

Table 3.5: Performance of our masked AES implementations.

shares with pre-loaded randomness. Note that all implementations are fully unrolled, so the code size can trivially be decreased to roughly a tenth when this is a concern. However, we do not expect this to be an issue for devices with a Cortex-A8 or similar microprocessors, as they are relatively high-end.

Following, we discuss how our implementation compares to related work. We note that one should be cautious when it comes to comparing cycle counts, in particular when benchmarks were obtained on different microarchitectures or from simulators. Analytically, despite the differences between ARMv4T and ARMv7-A, it is clear that we manage a noticeable improvement between the work of Goudarzi et al. [92] and ours. Comparing to Wang et al. [219], we require less randomness due to a different masking scheme and apply bitslicing instead of computing SubBytes with tower-field arithmetic. The Cortex-A15 is more modern and powerful than the Cortex-A8. It can decode 3 instructions instead of 2, has out-of-order execution, and its NEON unit has a 128-bit wide datapath instead of 64-bit. However, it has longer pipelines which means that the penalty for, for instance, wrong branch predictions will be higher. We ran their code on our Cortex-A8-based benchmarking device and measured 34,662 cycles for the 4-share implementation and 158,330 cycles for the 8-share implementation, but we cannot fully explain the difference due to the amount of possible causes and the unavailability of more detailed information. Comparing to Journault et al. [115] we show how the parallelism in SubBytes can additionally be exploited on a higher-end CPU with vector registers to improve performance. Compared to unmasked implementations, there is of course still a noticeable performance penalty for adding side-channel protections. The unmasked bitsliced AES implementation of Bernstein et al. [30] also exploits NEON to run at 19.12 cycles per byte (i.e., 306 cycles per block) in CTR mode, but that uses counter-mode caching and processes 8 blocks in parallel.

3.3.2 Side-Channel Evaluation of Higher-Order AES

This section provides a side-channel evaluation of the implemented masked AES cipher. Section 3.3.2.1 provides the experimental setup, Section 3.3.2.2 evaluates the

security order of the implementation and Section 3.3.2.3 expands the evaluation, using information-theoretic bounds.

3.3.2.1 Experimental Setup

Balasch et al. [13] described in detail how they performed DPA attacks on a BeagleBone Black running at 1 GHz. Our experimental setup and measuring environment follow their approach. The board is running Debian Jessie and has several processes running in the background. We power the board using a standard AC adapter and connect it to the measurement PC over Ethernet. A few lines of Python on the BeagleBone open a TCP socket and spawn a new AES process for every input that it receives. The measurement PC connects to the socket and sends inputs over Ethernet.

We use a LeCroy WaveRunner 8404M-MS oscilloscope with a bandwidth of 4 GHz, operating at a sampling rate of 2.5 GSamples/sec. The AES process sets a GPIO port high before the execution of AES and sets it low after AES is finished, so that it can be used as the trigger signal. We place a magnetic field probe from Langer, model RF-B 0.3-3, with a small tip on the back of the BeagleBone board, near capacitor 66. The probe is connected to a Langer amplifier, model PA 303 SMA. The acquired traces were post-processed using the Riscure Inspector toolchain in order to perform signal alignment. We note that OS-related interrupts in conjunction with time-variant cache behavior result in a fairly unstable acquisition process. Thus, the evaluator has to either discard a large portion of the acquired trace set or resort to more sophisticated alignment techniques such as elastic alignment [214]. The BeagleBone Black device is visible in Figure 3.9.

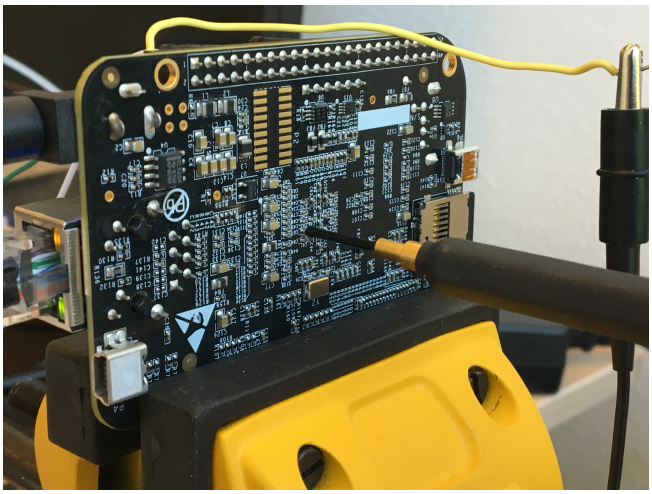


Figure 3.9: BeagleBone Black ARM Cortex A-8 and Langer RF-B 0.3-3

3.3.2.2 Security Order Evaluation

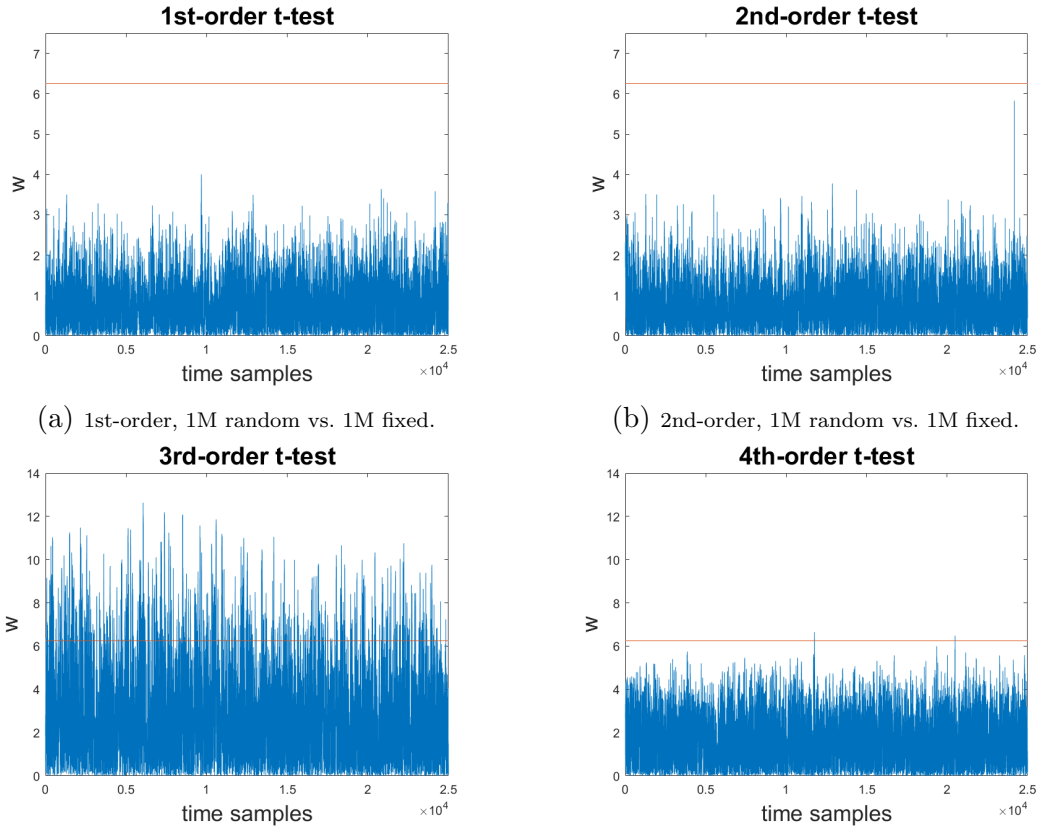
Since our implementation uses SNI gadgets, it maintains theoretical security against probing attacks of order $d-1$ or less. The natural starting point of our side-channel

evaluation is to identify any discrepancy between the theoretical and the actual security order, i.e., to determine the real-world effectiveness of the masking scheme. To achieve that goal, we need to assess whether the shares leak independently or whether the leakage function recombines them. Such recombinations can be captured by evaluating the security order in the bounded moment model [21] using, e.g., the leakage detection methodology [60, 185, 223].

What we encounter once again is the gap between the theoretical order of a masking scheme and its real-world counterpart [12]. Similarly to masked PRESENT, we face the issue of distance-based leakages, which can result in the order reduction of a scheme. Following the same pattern as Section 3.2, we evaluate the security order using the leakage detection methodology known as TVLA [60], which emphasizes detection over exploitation in order to speed-up the procedure. To make the evaluation feasible w.r.t. data complexity, we focus on the first round of our single-block 4-share implementation and employ the random vs. fixed Welch t-test, which uses random and fixed plaintexts acquired in a non-deterministic and randomly interleaved manner. Consecutively, we perform univariate t-tests of orders 1 through 4 using the incremental, one-pass formulas of Schneider and Moradi [185] at a level of significance $\alpha = 0.00001$. The results are plotted in Figure 3.10. Note that the number of samples per trace is fairly high due to the lengthy computation of the 4-share masked AES round and due to the high sampling rate dictated by the clock frequency (1 GHz) and the Nyquist theorem. As a result, the t-test methodology faces the issue of multiple comparisons and we need to control the familywise error rate using the Šidák correction $\alpha_{SID} = 1-(1-\alpha)^{1/\#samples}$ [188]. The leakage detection threshold th is then computed using the formula $th = CDF^{-1}_{\mathcal{N}(0,1)}(1-\alpha_{SID}/2)$, which equals to 6.25 when testing 25k samples per trace [223].

In Figure 3.10 we observe that for orders 1 and 2, a 1M random vs. 1M fixed t-test does not reject the null hypothesis, thus no leakage is detected in the first two statistical moments. The situation is different for higher orders: both the 3rd and the 4th-order univariate t-tests are able to detect leakage. This demonstrates that the actual security order of the implementation is less than the theoretical one and detecting the presence of 3rd-order leakage is in fact easier than detecting 4th-order leakage. Interestingly, the experimental results are not in direct accordance with the order reduction suggested by Balasch et al. [12], i.e., our 3rd-order (4-share) implementation achieves practical order of 2, while the theorized reduction suggests $\lfloor 3/2 \rfloor = 1$st-order security.

An additional way to approach the order reduction issue is to phrase it as a leakage certification problem [78,79]. The leakage certification procedure allows us to assess the quality of a leakage model w.r.t. estimation and assumption errors. Gauging the effect of estimation errors, i.e., those that arise from insufficient profiling, is straightforward and can be carried out via cross-validation techniques [81]. Assumption errors are more difficult to assess, since they arise from incorrect modeling choices and would ideally require the comparison between the chosen model and an unknown perfect model. To tackle this, the indirect approach of Durvaux, Standaert and Veyrat-Charvillon [79] observes the relation between estimation and assumption errors and if the latter are



(a) 1st-order, 1M random vs. 1M fixed.

(b) 2nd-order, 1M random vs. 1M fixed.

(c) 3rd-order, 1M random vs. 1M fixed.

(d) 4th-order, 1M random vs. 1M fixed.

Figure 3.10: Univariate leakage detection of orders 1 until 4.

negligible in comparison, they conclude that the chosen model is adequate.

In our approach, we use the t-test-based certification toolset of Durvaux et al. [78], which focuses on the assumption and estimation errors for each statistical moment. Initially, we start with an erroneous model for our 4-share implementation: we assume that the leakage is sufficiently captured by a Gaussian template, i.e., a normal distribution that is fully described by the first two statistical moments. The results are visible in the upper part of Figure 3.11, using a trace set of size 900,000. In particular, we plot the p-value of a t-test that compares an actual statistical moment (estimated from the trace set) with a simulated statistical moment (estimated by sampling the profiled model). A high p-value (i.e., a mostly white image) indicates that estimation errors overwhelm assumption errors and that the chosen model is adequate. A small p-value indicates that assumption errors are larger than estimation errors, thus the chosen model is erroneous. The process is repeated for all first four statistical moments (mean, variance, skewness, kurtosis) using cross-validation.

In the first two images of Figure 3.11 (upper part, mean and variance), the high p-values indicate that these moments are well-captured by the model. Naturally, the fourth image (upper part, kurtosis) is black, indicating that the model disregards the 4th moment of a parallel 4-share implementation which should (in theory) contain useful information. Interestingly, the third image (skewness) is also black, penalizing

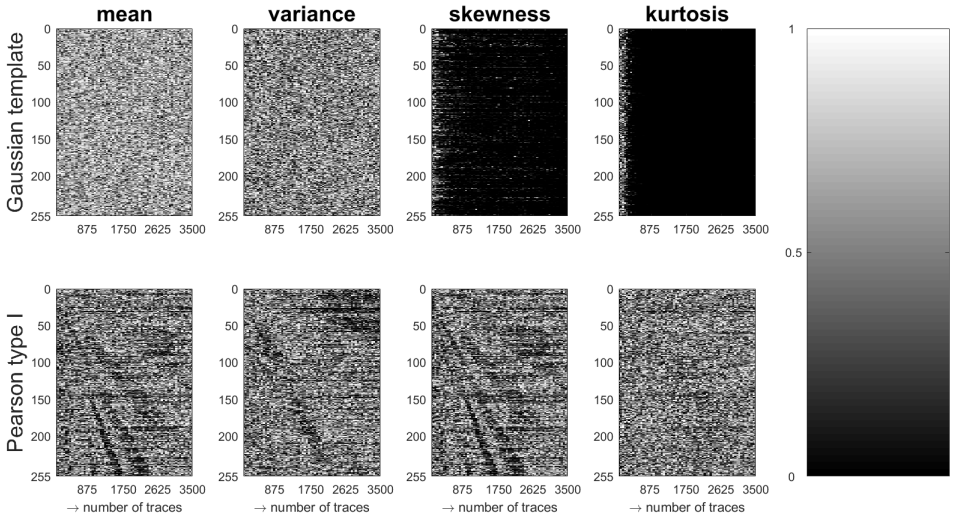


Figure 3.11: Leakage certification p-values for Gaussian templates and Pearson type I.

any model that does not include the 3rd statistical moment, although in a perfect scheme it should not convey any information. We continue this approach with a more adequate model for the 4-share implementation: we assume that the leakage is captured by a Pearson type I distribution [186], i.e., a 4-moment Beta distribution. The results are visible in the lower part of Figure 3.11 and show that the assumption errors in the 3rd and 4th moments tend to be smaller than the corresponding estimation errors.

As demonstrated by both the t-test methodology and the leakage certification process, the NEON-based implementations on ARM Cortex-A8 are likely to be subject to order reduction and may require further hardening to prevent dependencies between shares. The potential causes of the order reduction remain unexplored since they may stem from bus/register/memory transitions, pipelined data processing or even electrical coupling effects. Pinpointing the origin of the security reduction remains an open problem in the side-channel field since it essentially requires the countermeasure designer to access/modify the hardware architecture and chip layout, a task that is not possible with proprietary designs.

3.3.2.3 Information-Theoretic Evaluation

Having investigated the security order of the single-block 4-share AES implementation, we turn to the evaluation of its 8-share counterpart. The core feature of a masking scheme is the noise amplification stage. Assuming sufficient noise, it has been shown that the number of traces required for a successful attack grows exponentially w.r.t. the order $d-1$ [49]. As a result, the evaluation of the proposed 8-share implementation can be beyond the measurement capability of most evaluators. To tackle this issue, we will rely on an information-theoretic approach used by Standaert et al. and Journault et al. [115, 197, 200], assisted by the bound-oriented works of Prouff and Rivain [170], Duc, Faust, and Standaert [76], and Grosso and Standaert [98].

Analytically, we start with an unprotected (single-share) AES implementation and

estimate the device/setup signal to noise ratio (SNR). We define the random variable S to correspond to the sensitive (key-dependent) intermediate values that we try to recover. Likewise, we define the random variable L to correspond to the time sample that exhibits high leakage (heuristically chosen as the sample with the highest t-test value). Subsequently, we profile Gaussian templates for all sensitive values s that are instances of variable S. In other words, we estimate $\hat{Pr}[L|s]_{model} \sim \mathcal{N}(\hat{\mu}_s, \hat{\sigma}_s^2)$ for all s. Using the estimated moments, we compute the SNR as the ratio $\hat{var}_s(\hat{\mu}_s)/\hat{E}_s(\hat{\sigma}_s^2)$, resulting in SNR ≈ 0.004. We continue to compute the Hypothetical Information (HI) which shows the amount of information leaked if the leakage is adequately represented by the estimated model $\hat{Pr}_{model}$.

$$\mathrm{HI}(S;L) = H[S] + \sum_{s\in\mathcal{S}} Pr[s] \cdot \int_{l\in\mathcal{L}} \hat{Pr}_{model}[l|s] \cdot log_2 \hat{Pr}_{model}[s|l] \;\; dl,$$

$$\text{where } \hat{Pr}_{model}[s|l] = \frac{\hat{Pr}_{model}[l|s]}{\sum_{s^*\in\mathcal{S}} \hat{Pr}_{model}[l|s^*]}$$

To simplify the evaluation process, we employ the independent shares' leakage assumption so as to extrapolate the information of a single share to the information of a d-tuple of shares. Thus, in order to obtain the HI bounds for security orders 3 and 7, we raise $\mathrm{HI}(S;L)$ to the security order. In addition, the evaluator should take special consideration w.r.t. horizontal exploitation [25, 215], which can be particularly hazardous, e.g., in the context of lengthy masked multiplications. To showcase such a scenario, we employ the bound of Prouff and Rivain [170], stating that the multiplication leakage is roughly $1.72d + 2.72$ times the leakage of a d-tuple of shares. The results of the information-theoretic evaluation are visible in Figure 3.12.

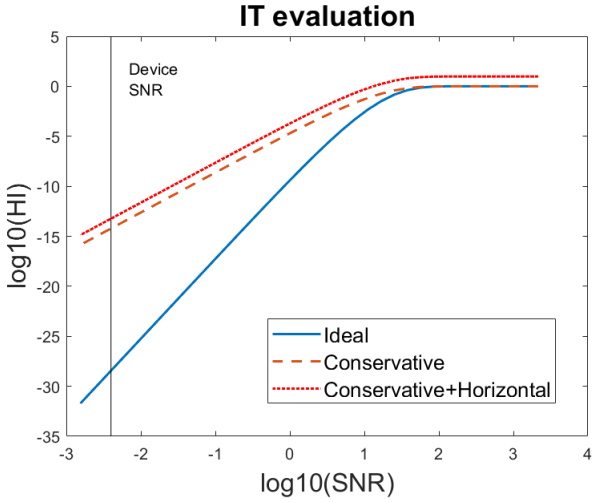


Figure 3.12: Information-theoretic evaluation for the 8-share masked implementation.

Figure 3.12 assesses the performance of the proposed 8-share AES implementation, using information-theoretic bounds. The solid line shows the ideal masking performance, while the dashed line shows a conservative masking performance due to order

reduction from order 7 to order 3. Last, the dotted line demonstrates the scenario where the adversary exploits the order-reduced (conservative) version in a horizontal fashion, i.e., he incorporates all intermediate values computed during a masked AES multiplication. For the current SNR of the device, the measurement complexity is bounded by approximately 2^{91} measurements (ideal case), 2^{45} (conservative case) and 2^{42} (conservative horizontal case) [76].

3.4 Bridging the Gap

Through side-channel evaluations, Sections 3.2 and 3.3 demonstrated the hazardous gap between masking theory and practice. In particular, they showcase experimentally that designing a masking scheme using the value-based leakage model can result in order reduction and poor performance once deployed in a real-world device. This detrimental gap is observed in multiple devices such as low-end AVRs, as well as midrange and high-end ARM architectures. Thus, Section 3.4 works toward mitigating it. In particular, Section 3.4.1 describes the experimental setup. Section 3.4.2 examines such order-reducing effects in an ATMega163 smartcard. Despite being a fairly outdated device, ATMega163 is ideal to investigate such effects due its particularly low noise levels. This enables us to experimentally test several ILA-breaching effects and identify their origin, with little worry about such effects being "buried" in noise and remaining undetected. The accumulated knowledge leads to Section 3.4.3, where we develop a "hardened" 1st-order implementation of the RECTANGLE Sbox. The "hardened" Sbox is capable of resisting ILA-breaching effects and maintain its security order.

3.4.1 Experimental Setup

Our setup uses the AVR ATMega163 microcontroller, clocked at 4MHz. The traces are captured with a Picoscope 5203 at sampling rate of 25 MSamples/sec and are post-processed using static align. The Riscure power tracer, enables us to produce a trigger signal using the smartcard IO pins. The setup can be seen in Figure 3.13.

3.4.2 ILA-Breaching Effects

In this section, we present three effects identified in the ATMega163 microcontroller that breach ILA and pose a hazard to any masking scheme's security. Analytically, the effects below demonstrate that independent computations *do not* necessarily lead to independent leakages and thus, the order-reduction theorem can become applicable. Every effect (Sections 3.4.2.1, 3.4.2.2 and 3.4.2.3) is described as a standalone, assembly-based scenario that manipulates two 4-bit shares x_0, x_1 originating from the sensitive, key-dependent, 4-bit value x, such that $x = x_0 \oplus x_1$. The shares x_0, x_1 are always manipulated in a theoretically sound manner, adhering to the masking scheme's requirements, i.e. we never combine the shares directly (e.g. via an exclusive-or instruction `eor x0, x1`).

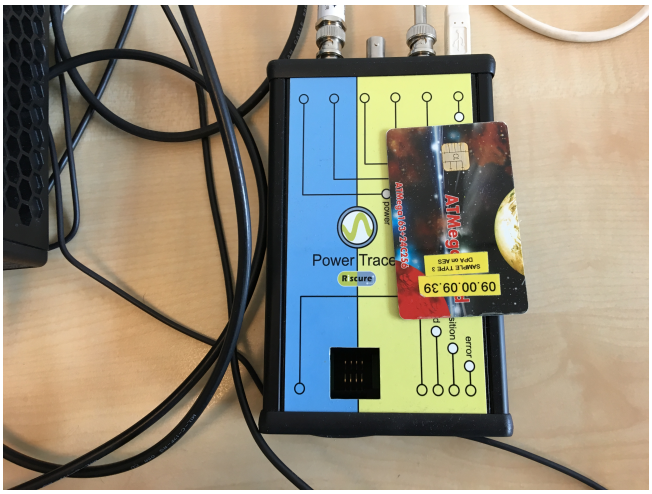


Figure 3.13: ATMega163 smartcard and Riscure power tracer.

For all the described scenarios, that are theoretically sound, we show experimentally that ILA is not fulfilled by employing 1st-order, univariate techniques. Namely, we perform correlation-based analysis [41], computing the correlation coefficient ρ between the Hamming weight of the sensitive, key-dependent value x and the experimentally acquired traceset. To maintain a wide attack scope, we also use the leakage detection methodology [60,185] and compute the 1st-order, random vs. fixed t-test. We conclude every scenario by suggesting possible solutions that enforce ILA. Restating Balasch et al. [12], as we are always limited by the traces at hand, we cannot rule out the existence of 1st-order leakages, yet we establish that their informativeness is substantially limited compared to 2nd-order leakages in the target device. Note that extra care is taken in order to assess all effects independently, i.e. we use the suggested solutions in order isolate the effect under discussion from the rest.

The analyzed effects can manifest in several data storage units (e.g. registers, SRAM/Flash memory cells, I/O buffers, etc.) and may relate to different instructions of the AVR ISA[8], leading to a very large number of potential scenarios. In order to maintain a feasible scope, we limit our discussion to storage units and instructions that are often encountered in the context of cryptographic implementations, i.e. SRAM memory accesses (`ld, st`) and logical instructions (`eor, and, or`).

3.4.2.1 Overwrite Effect

The overwrite effect is observable when a share gets overwritten by a different share from the same family. For instance, if share x_0 in a data storage unit (register, memory cell, etc.) gets overwritten by share x_1, then the power consumption correlates with the number of bits switched i.e. $x_0 \oplus x_1$. This effect was observed by Daemen et al. [208] and later revisited by Coron et al. [63].

Below, we address the most common situations in which overwriting arises during a cryptographic implementation. We perform two experiments: a register-based

[8]http://www.atmel.com/images/Atmel-0856-AVR-Instruction-Set-Manual.pdf

overwrite via the instruction `mov x0, x1`, and a memory-based overwrite via the instruction `st SRAM_x0, x1`. The experiments are described in Listings 3.1 and 3.2. Their analysis follow in Figure 3.14.

We confirm that overwriting is indeed an ILA-breaching effect, manifesting both in registers and SRAM memory. Note that the exploitability of the effect varies according to the data storage unit: in ATMega163, register-based overwriting can be exploited with roughly 500 traces (3.14a), while memory-based requires at least 40k traces (3.14c). Preventing register and memory-based overwrites is straightforward: the corresponding register (or memory cell) needs to be cleared in advance.

Listing 3.1: Register overwrite experiment.

```
;share x0 in r17
;share x1 in r23
mov r17,r23
;
;
```

Listing 3.2: Memory overwrite experiment.

```
;share x0 in SRAM 0x0080
;share x1 in r17
ldi r27,0x00
ldi r26,0x80
st X,r17
```

3.4.2.2 Memory Remnant Effect

The memory remnant effect is a leakage effect originating from consecutive SRAM accesses to shares of the same family. Assume that shares x_0, x_1 are stored in SRAM cells and get accessed sequentially. Naturally, the first access leaks share x_0 (value-based leakage), yet it also creates a "remnant" of x_0. The second access will leak the transition of the share x_1 and the remnant x_0, reducing the security.

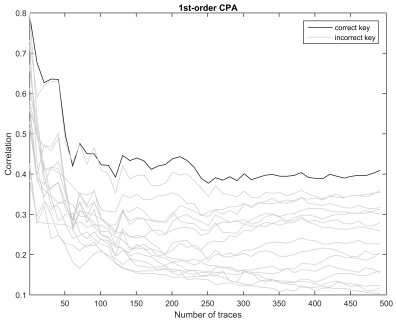
Listing 3.3: Memory remnant experiment.

```
;share x0 in 0x0080
;share x1 in 0x0090
ldi r27, 0x00
ldi r26, 0x80
ld r17, X
ldi r27, 0x00
ldi r26, 0x90
ld r20, X
;
;
```

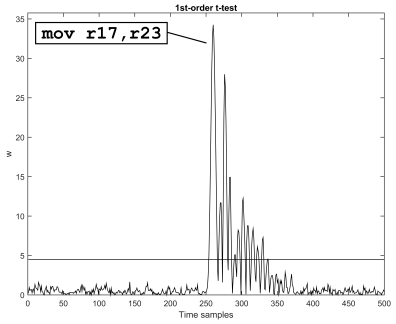
Listing 3.4: Clearing remnant experiment.

```
;share x0 in 0x0080
;share x1 in 0x0090
ldi r27, 0x00
ldi r26, 0x80
ld r17, X
ldi r17, 0x00
ldi r26, 0x85
ld r17, X
ldi r26, 0x90
ld r20, X
```

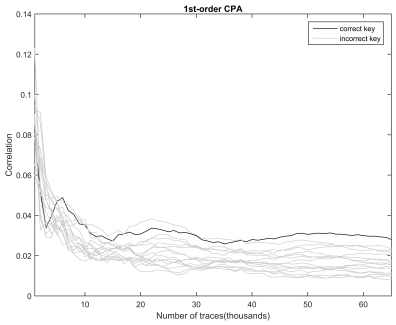
We address the remnant scenario with two experiments. Listing 3.3 demonstrates how two consecutive SRAM accesses `ld rA, SRAM_x0`, followed by `ld rB, SRAM_x1` produce the remnant effect. Second, in Listing 3.4, we show how clearing the register and accessing an unrelated SRAM address (`0x0085`) can remove the remnant.



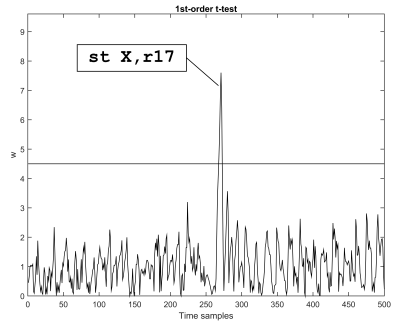
(a) Register overwrite, 1st-order CPA, HW model, 500 traces.



(b) Register overwrite, 1st-order t-test, 5k random vs. 5k fixed.



(c) Memory overwrite, 1st-order CPA, HW model, 65k traces.



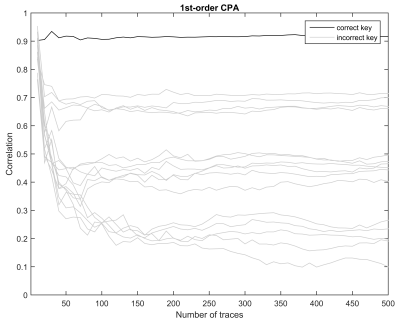
(d) Memory overwrite, 1st order t-test, 50k random vs. 50k fixed.

Figure 3.14: Register/memory-based overwrite effects

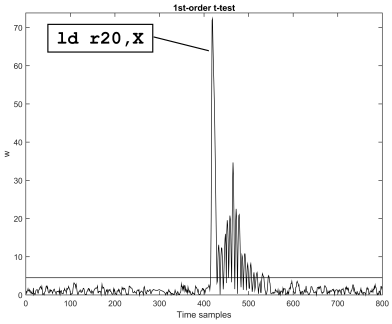
As shown in Figures 3.15a and 3.15b, consecutive SRAM accesses can potentially lead to ILA violations. Exploiting (in a univariate manner) the memory remnant effect in ATMega163 needs less than 500 traces with our setup. Preventing the effect requires the clearing of the register and the insertion of a dummy SRAM access. Alternatively, the implementor could ensure that same-family shares are not accessed sequentially. Note also that the `st` instruction produces a similar effect. We speculate that the memory remnant effect is caused by the structure of the the memory access mechanism and potentially, the pipelining stages.

3.4.2.3 Neighbour Leakage Effect

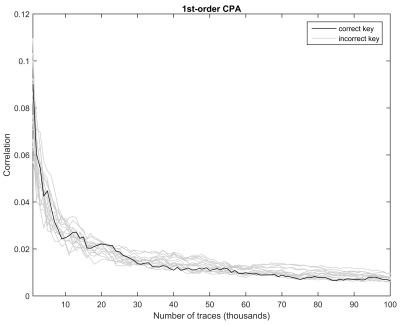
The neighbour leakage effect implies that accessing or processing the contents of a data storage unit will cause leakage in another unit as well. For example, assume that share x_0 is stored in register `rB` and share x_1 is being processed in register `rA`. Assume also that the registers `rA, rB` are subject to the neighbour leakage effect. Processing `rB` will produce a value-based leakage of x_0. At the same time, the neighbouring leakage effect will cause `rA` to leak the value of x_1, resulting in transition between shares and the recovery of sensitive value x. The following two experiments (Listing 3.5) verify the neighbour leakage effect between registers `r2, r3`, i.e. a share stored in `r2` leaks



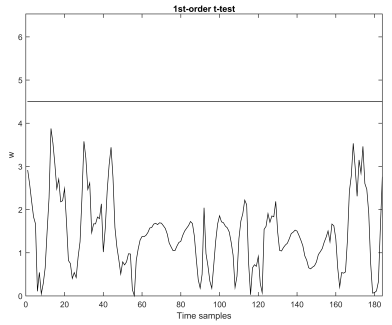
(a) Memory remnant effect,1st-order CPA, HW model, 500 traces.



(b) Memory remnant effect, 1st-order t-test, 5k random vs. 5k fixed.



(c) Clearing remnant effect,1st-order CPA, HW model, 100k traces.



(d) Clearing remnant effect, 1st-order t-test, 100k random vs. 100k fixed.

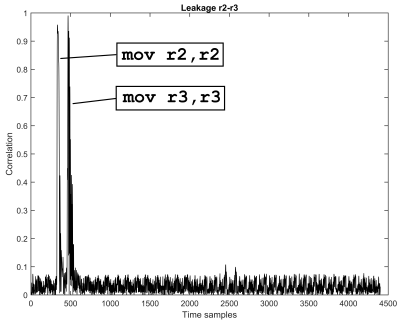
Figure 3.15: Memory-based remnant effect

when manipulating `r3` and vice-versa.

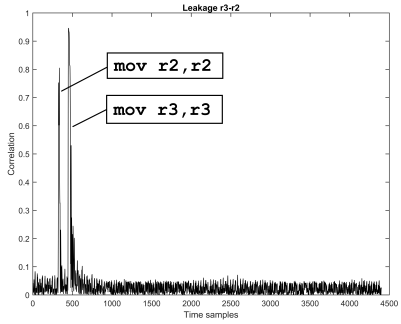
Listing 3.5: Neighbour leakage experiment for `r2` and `r3`.

```
; clear all registers
; sensitive 'x' is in the selected register (r2 OR r3)
mov r0, r0
nop ; 5 times
mov r1, r1
nop ; 5 times
mov r2, r2
nop ; 5 times
mov r3, r3
nop ; 5 times
...
mov r31, r31
```

As shown above, we use the same code from Listing 3.5, but in the first time we



(a) Correlation $\rho(HW(x_0), traceset)$, `r2-r3`, 5k traces.



(b) Correlation $\rho(HW(x_0), traceset)$, `r3-r2`, 5k traces.

Figure 3.16: Neighbour-based leakage effect

put the sensitive variable x into register `r2` (*only* line `mov r2, r2` should result in leakage). In the second time, we put the sensitive value into the register `r3` (*only* line `mov r3, r3` should leak). However, Figure 3.16 shows that both register accesses leak. As a result, we have identified a pair of data storage units (`r2,r3`) that exhibit the neighbour leakage effect. Note that in this case the effect is symmetrical, i.e., `r2` triggers `r3` and vice-versa (Figures 3.16a and 3.16b). We also observed that the effect is persistent, i.e. the `mov` instructions will trigger the same behavior, even if performed later (not necessairly in order as in Listing 3.5). We run the same experiment in order to identify all possible neighbour leakages in the register file (all pairs in set {`r0,...,r31`}). The results are available in matrix R. The 32x32 matrix R is generated experimentally, while investigating all possible neighbouring leakage effects in the ATMega163 register file (by performing 32 experiments similar to Listing 3.5). Value '1' denotes the presence of leakage and '0' the absence. We can see that the issue mostly affects consecutive registers, although exceptions exist, e.g. register `r0`. We did not identify a similar effect in SRAM memory, yet our experiments were limited to a small region of cells. Neighbour-like effects have been observed in consecutive instructions, yet it remains open whether they are cause by proximity or they stem from other effects. We speculate that they relate to the structure of the register file and likely involve the storage and multiplexing mechanism of the registers. Given the pairwise manifestation of the effect, we speculate a pair-based organization of the register file. Still, note that it is hard to link architectural options at the hardware layer directly to side-channel effects, a situation that persists in Chapter 3 and will resurface in Chapter 6. As a solution to the neighbour effect, the developer can opt to avoid storing shares in hazardous registers and keep a safety distance between consecutive instructions. Alternatively, he can store all shares in SRAM, except for the ones currently in use.

Summing up, we stress the following focal points regarding the ILA-breaching effects and their solutions:

- All identified effects are device-dependent, i.e. there is no hard guarantee that they are observable and reproducible in different AVR-based microcontrollers,

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
00	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
01	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
02	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
09	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**

Table 3.7: Neighbouring leakage effects in the ATMega163 register file.

let alone different architectures such as ARM, TI, PIC etc. Both intra-AVR and inter-architectural observability of the effects remains open.

- The effects are often counter-intuitive when viewed in the assembly layer of abstraction. They originate from the hardware and/or the physical layer, thus can only be detected via experimental evaluation. Linking the assembly ILA-breaching effects to a particular hardware component or physical phenomenon is non-trivial [179, 203], especially without knowledge of the underlying chip architecture and properties.

- Since the effect's detection requires experimental evaluation, different instructions or code arrangements can potentially lead to additional, unidentified ILA-breaching effects. Still, we maintain that it is possible to construct "hardened" masked operations in ATMega163 by removing the identified effects (see Section 3.4.3). It remains open whether the suggested solutions are computationally optimal or more efficient clearing techniques can be identified.

The takeaway message of this section is that assembly-level soundness cannot enforce ILA and hence 1st-order security, due to the nature of the breaching effects. However, it is possible to acquire sufficient knowledge about effects and solutions in a particular device. These non-intuitive checks discussed above can be subsequently integrated into "hardened" assembly code.

Table 3.8: Masked Sbox comparison in ATMega163

Order d	**Hardened**	**Latency cycles**	**Throughput bits/cycle** $\times 10^{-3}$	**RNG bytes**
Unprotected	no	32	250	0
1st order	no	87	91	4
	yes (eff.)	993	8	4
	yes (cons.)	1319	6	4
2nd order	no	775	10	12

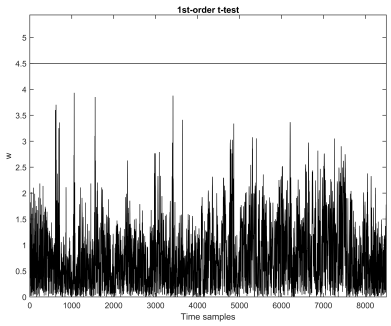
3.4.3 Harderned Implementation & Side-Channel Evaluation

The current Section builds up on the advances of Section 3.4.2 by putting forward a "hardened", 1st-order masked, ISW-based RECTANGLE Sbox. The desired aim is to produce an assembly-based, lightweight Sbox implementation that is secure against 1st-order, univariate attacks, hence forcing the attacker to resort to 2nd-order techniques.

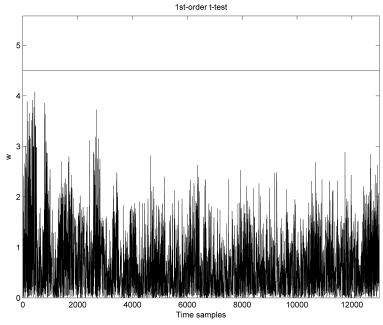
Our implementation opts for a bitsliced [32, 66] representation, due to both the bitsliced structure of RECTANGLE and to the $GF(2)$-oriented nature of the ISW countermeasure. We employ a bitslicing factor of 2, i.e. we exploit the 8-bit AVR architecture in order to process two 4-bit Sboxes in parallel (nibble-slicing). The Sbox is decomposed into $GF(2)$ operations which can be accelerated by via SIMD-like, 8-bit assembly instructions. The decomposition suggested by Zhang et al. [224] is optimal w.r.t. $GF(2)$ multiplicative complexity, looking again in the work Grosso et al. [95], which established that the minimum number of non-linear operations required by 4x4 Sboxes is 4.

In order to "harden" the Sbox, we use the solutions suggested in Section 3.4.2 and follow two approaches: efficient and conservative. In the *efficient* approach, after processing any share, we clear the registers on a need-to basis and insert dummy `ld` instructions to avoid overwrite and remnant effects. We avoid neighbouring leakage effects by always storing the shares in SRAM, i.e. the register file contains only the shares used by the current instruction. In the *conservative* approach, we perform all the afore-mentioned clearing techniques. In addition, we insert dummy `st` instructions and perform thorough register/memory clearing. Both efficient and conservative approaches are applied to every single instruction of the implementation, i.e. the cost is linear w.r.t. the number of instructions that manipulate masked shares. The resulting computational overhead is significant: the efficient "hardened" Sbox implementation runs in 993 clock cycles, i.e. almost 12 times slower compared to the "naive" 1st-order, ISW-based RECTANGLE Sbox, which runs in 87 clock cycles. The conservative "hardened" Sbox implementation requires 1319 clock cycles, i.e. it is 15 times slower. Table 3.8 contains a comparison between "naive" 1st-order, "naive" 2nd-order and efficient/conservative "hardened" 1st-order bitsliced implementations of the RECTANGLE Sbox in AVR assembly.

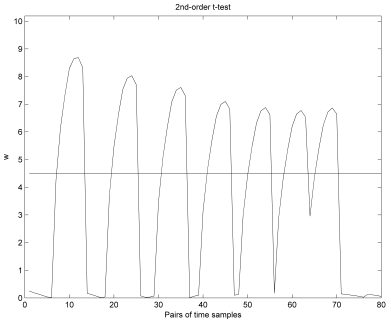
Using the random vs. fixed t-test, we evaluate the efficient and conservative "hardened" 1st-order Sboxes, as well as the "naive" 1st-order Sbox. Using a 25k



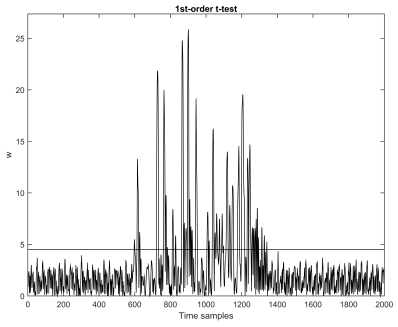
(a) Efficient hardened Sbox, 1st-order t-test, 25k random vs. 25k fixed.



(b) Conservative hardened Sbox, 1st-order t-test, 100k random vs. 100k fixed.



(c) Consevative hardened Sbox, 2nd-order t-test, 25k random vs. 25k fixed.



(d) Naive Sbox, 1st-order t-test, 1k random vs. 1k fixed.

Figure 3.17: Hardened and naive Sbox evaluations

random vs. 25k fixed t-test does not yield any statistically significant leakage in the efficient "hardened" version (Figure 3.17a). However, we note that a 50k random vs. 50k fixed t-test is able to detect leakage, i.e. trying to reduce the cost of enforcing ILA can have a detrimental effect on security. For the conservative "hardened" Sbox, a 100k random vs. 100k fixed t-test does not detect any leakage (Figure 3.17b). Note that a *2nd-order* 25k random vs. 25k fixed t-test on a chosen sample window is able to detect leakage. Therefore, we conclude that for the given device, the informativeness of 1st-order attacks is substantially limited and a 2nd-order attack is the preferable adversarial strategy (Figure 3.17c). Naturally, the "naive" 1st-order version rejects the null hypothesis (Figure 3.17d) due to the ILA-breaching effects and the 1st-order leakage can be easily exploited.

So far, the only way to guarantee the actual security order of a real-world implementation was to increase the scheme's theoretical order d, in order to ensure that the implementation attains an actual order of $\lfloor \frac{d}{2} \rfloor$ [12]. Clearing the ILA-breaching effects requires a significant overhead and is device-dependent, yet it is the *only* technique known to us that can enforce 1st-order, univariate security. In addition, hardening does not increase the scheme order d, thus the RNG cost is not increased. The previous suggestions require a higher scheme order, hence a significant overhead, since both the implementation cost and the RNG cost are quadratic w.r.t. the order.

We compare the "hardened" 1st-order and "naive" 2nd-order implementation costs (in clock cycles) and we observe that hardening the 1st-order Sbox is slower than increasing the scheme's order from 1 to 2 (both in the efficient and in the conservative case). Still, the solution requires no extra RNG and we maintain that removing these effects can also be beneficial to higher-order implementations, i.e. it is complimentary to masking. The extent to which higher-order implementations can benefit from removing ILA effects remains an open problem. Similarly, the exact cost/quality of RNG is unclear in many applications; pseudo-RNG can imply the optimistic cost of a two-round AES execution per 16 random bytes or the pessimistic cost of a ten-round AES execution per 16 random bytes [96]. As a result, a 2nd-order implementation AVR requires 18k cc (optimistic) or 93k cc (pessimistic) [5] for a full RECTANGLE execution[9], while a hardened 1st-order implementation requires only 6k cc (optimistic) or 31k cc (pessimistic)[10]. We will revisit the standalone cost of RNG in Chapter 4, where we will also investigate its interactions with SCA security.

3.5 Conclusions & Future Directions

In this chapter, we investigated the speed and space requirements of a bitsliced implementation of PRESENT on the ARM Cortex-M4 architecture, protected with 2nd-order ISW masking. In addition, we have shown how higher-order masking of AES can be speeded up using NEON vector registers of ARM Cortex-A8. In both ARM architectures, we confirm experimentally the order-reduction effects. Regarding future work, we note that the optimal strategy to attack masked implementations remains open. The amount of leakage available in various security orders and the lack of transparency when it comes to serial and parallel processing of shares makes a fair and concrete evaluation difficult. For instance, using a higher-order univariate methodology (like in masked AES), implicitly assumes that all the shares are manipulated in parallel. While this appears to hold when looking at the NEON assembly specifications, full parallelism may not be enforced on a hardware level. A deeper inspection of the circuitry could potentially clarify the actual parallelism and guide us towards more effective attacks. Moving towards multivariate exploitation, practical horizontal attacks such as soft-analytical attacks need to be carried out such that we can gauge in practice the detrimental effects of lengthy leaky computations and establish an even more solid evaluation procedure. Finally, from the defenders point of view, this chapter focused on the computational cost of masking, largely ignoring the RNG cost implied (with the exception of the RNG improvements of Section 3.3.1.1. Since RNG can pose a large bottleneck, Chapter 4 will investigate it in larger detail.

Moreover, this chapter investigated the hazards in software masking and established a secure, 1st-order masked Sbox implementation against 1st-order, univariate attacks. Still, several important questions for future work arise due to this effort. We demonstrated that removing the ILA-breaching effects is feasible, yet identifying the best clearing mechanism and minimizing the overhead is a topic for further exploration.

[9] 25 rounds, 4 16-bit AND per round, 3 16-bit random numbers per AND

[10] 25 rounds, 4 16-bit AND per round, 1 16-bit random numbers per AND

Similarly, the current work is limited to AVR ATMega163 and needs to be extended to different devices and platforms. Finally, we observe that the effects identified depend on the architecture and the physical layer, making the assembly layer an error-prone abstraction layer to work at. Future work can strive towards custom-made architectures that enforce ILA in hardware and result in predictable assembly instructions that do not compromise security. Ideally, such a construct should be able to guarantee ILA directly through careful design, without relying on additional countermeasures such as threshold implementations.

Chapter 4

Recycling Randomness

"Random number generation is too important to be left to chance."

Robert Coveyou, 1970

Ostensibly, the last decade of side-channel research has seen the rise and establishment of masking and its glitch-resistant counterpart, threshold implementations as the *de facto* side-channel countermeasures. Masking's multifaceted forms have emerged in software and hardware, prompting research on its security properties [111]. Concurrently, the tweakable masking representations enabled high-performance implementations on a multitude of platforms and devices, similar to those presented in [70, 94]. Finally, the countermeasure sparked in-depth laboratory evaluations, new attack pathways and, naturally, discussions about its effectiveness like order-reducing effects. Chapter 3 serves as a testament to the popularity of this countermeasure, while it pushes the performance limits and attempts to fill in the gaps between theory and practice. Following a similar trajectory to masking, shuffling a cipher's operations has also established approval within the hardware security community, leading to multiple countermeasure implementations [181] and new exploitation techniques [216].

A unifying feature between masking and shuffling countermeasures is their inherent reliance on random number generation. Masking requires random numbers to "hide" the values of intermediate values, while shuffling needs random numbers to permute the cipher's operations. Broadly speaking, the research field of random number generation has been actively proposing generators in various forms (true, deterministic, hybrid generators) [17] and kept examining the statistical properties of the generated random stream. Nonetheless, customized RNG for masking and shuffling is often left unexplored, despite the symbiotic link between these countermeasures and random numbers. So far, a large part of the community, including the work in Chapter 3, was geared towards efficient design and implementation of the countermeasure itself, dismissing the large performance overhead that stems from RNG peripherals. As a result, the scheme's computational overhead kept decreasing, while the respective RNG overhead persisted or just remained in a state of flux. Advances in the area of RNG for masking and shuffling have also been partially hindered by the lack of communication between the academic and industrial sectors. This situation has limited the availability

of industrial-grade, open-source generators and has also enhanced the obscurity in several random number generators of industrial devices.

Propitiously, recent academic work sheds more light on the necessity and impact of random number generation on side-channel countermeasures [85]. This chapter is based on work published in [160] and is motivated by the need to study the impact of RNG on countermeasures, both in terms of security and in terms of performance. When originally stating that "random number generation is too important to be left to chance", the core consideration was *how* do we construct sound and secure generators. Interestingly enough, this chapter shall demonstrate that the phrase retains its merit when considering *how often* do we use the afore-mentioned generators. The main points of the chapter are summarized below.

- This chapter reconsiders how often do we need to perform random number generation for masking and shuffling and proposes efficient countermeasure variants. The security of the novel variants is extremely flexible and can be adapted based on the device's RNG capabilities and the designer's choices.

- This chapter marks a shift from the traditional three-stage process of designing, implementing and evaluating a countermeasure and signifies a different take on side-channel protection. Instead of viewing a countermeasure as a set-in-stone protection mechanism (that we need to optimize a priori), we opt to view it a an integrated component which the designer can mold and tweak according to his security needs. In this adaptive process the random number generator acts like a fine-tuning instrument which enables the designer to customize the countermeasure and adapt it to any application context.

4.1 Introduction

Following the work in Chapter 3, it is easy to see that when implementing masking or shuffling, the RNG requirements impose a non-negligible performance overhead that can impact the latency/throughput of the cryptographic implementation. Even if we avoid such impact by performing the RNG procedure during idle phases of the device, the required computations will directly increase the device's energy consumption. In order to meet the cumbersome RNG needs of masking and shuffling, designers can employ several options with respect to the generator type. For instance, they can opt to use a symmetric cipher structure as a pseudo random number generator, relying on either a full-round or a reduced-round block cipher [96] or on a stream cipher construct. Alternatively, they can opt directly for physical RNG or for structures that use a physical random number generator to seed deterministic random bit generators [17]. Recently, Faust et al. [85], has shifted the focus from constructing an efficient RNG, towards decreasing the required RNG cost in total. In particular, they have introduced the concept of amortizing randomness in a masking scheme, i.e. *recycling* the available randomness between several gadgets in order to reduce the RNG cost. Their work establishes the notion of security with common randomness (denoted as $t-$SCR) and provides composable ($t-$SNI) gadgets [19] that achieve randomness recycling. However, their analysis relies on simulation-based proofs that do not take into account the effect of recycling on the noise level of the device and on the noise amplification stage of masking. Naturally, this has lead to further investigation of randomness recycling and randomness reduction in general, motivating the work in this chapter.

4.1.1 Chapter Contribution

In this chapter, we put forward low-randomness versions of standard masking and shuffling countermeasures, which we refer to as Recycled Randomness Masking and Reduced Randomness Shuffling respectively. RRM and RRS are able to reduce the RNG overhead by employing memory units to store random numbers and reuse them later, e.g. in subsequent executions of the protected cipher. We provide several efficient $t-$NI custom multiplication gadgets for low-order RRM schemes that reduce randomness in an efficient manner. Similarly, we provide several design patterns that can reduce randomness when shuffling. In both cases, trading RNG overhead for memory overhead implies that every random number that is reused needs to be stored. As a result, the proposed RRM and RRS schemes are geared towards microcontroller units and possibly high-end FGPAs, since such devices can offer a fairly large amount of memory storage. On the contrary, the more strict area requirements in ASIC devices, encourage recycling on-the-fly, similarly to Faust et al. [85].

Subsequently, we investigate the noisy leakage security of RRM and RRS. We note that the formal approach of Faust et al. [85] has already investigated the $t-$SNI property of certain RRM schemes, introducing $t-$SCR. In order to establish a more holistic notion of security, we complement their approach by performing an analysis of a $t-$NI RRM scheme under the noisy leakage model, using the MI framework for

SCA [197]. In particular, we demonstrate how reducing the available randomness for performance/cost reasons interacts with the noise amplification stages of RRM and RRS. Thus, we establish a direct link between the randomness cost and the noisy leakage security level provided by a countermeasure, i.e. *we integrate the noise factor in our analysis*. We conclude that this randomness-security tradeoff constitutes a potent tool in the designer's arsenal that enables us to provide adequate security (in the noisy leakage model), while limiting the computational cost that stems from RNG.

4.1.2 Previous Work

Multiplying two families of shares under an ISW Boolean masking scheme consists of the computation of all partial products, as well as a compression algorithm that produces the final result, while injecting randomness [28,111]. Several implementation techniques and evaluation strategies have been suggested in the context of masking. With respect to implementation aspects, the techniques proposed range from lookup-table techniques [61,219] that re-randomize memory tables to GF-based circuits [45,92,180] that generate and inject randomness in several gadgets.

Regarding the evaluation strategies for masking, this chapter relies heavily on the recent advances by Battistello et al. [25] and Grosso et al. [98]. These works suggest that masked multiplications are prone to horizontal attacks, i.e. attacks that exploit several noisy intermediate values that are computed during the scheme. In this chapter, we put specific emphasis on the impact of horizontal exploitation to the noisy leakage security level of the scheme.

In the application of Boolean masking schemes, secure multiplications require quadratic data complexity w.r.t. randomness, in order to ensure the refreshing of partial products. Initially, Rivain et al. [180] extended the ISW scheme [111] and put forward a d-private compression algorithm (RP) that can compute dth-order secure multiplications in $GF(2^n)$ using $d(d+1)/2$, n-bit elements. Following, Belaïd, Benhamouda, Passelègue et al. [28] suggested an improved d-private compression (BBP) that performs partial product refreshing using $\lfloor d^2/4 \rfloor + d$ random numbers for security orders $d > 4$. In addition, they derived optimal compression algorithms for security orders $d = 2, 3$ and 4 which have data complexity, respectively $2, 4$ and 5 random elements per multiplication.

Despite recent efforts, it is notable that high-order masking implies a severe RNG overhead. Making for instance a 2nd-order secure AES implementation with optimal compression (2nd-order secure BBP) requires 10240 random bits per block encryption[1]. Generating this amount of random bits with a pseudo AES-based random number generator in ATmega microcontrollers implies an optimistic cost of roughly 20k clock cycles (2-round AES generator) and a pessimistic cost of 100k clock cycles (10-round AES generator) [5, 96]. We observe that the pessimistic case is fairly close to the computational cost of the 2nd-order secure AES on AVR devices [11], i.e. it amounts to approximately 38% of the clock cycles. Similarly, a 2nd-order secure PRESENT

[1]The cipher runs for 10 rounds consisting of 16 Sboxes, each requiring 32 $GF(2)$ multiplications that need 2 random elements for refreshing the partial products.

implementation on an ARM Cortex-M device spends 25% of its execution time for TRNG, as shown in Chapter 3, Section 3.2. The severe overhead of RNG in masking countermeasures can render the implementation cost prohibitive for small embedded devices and has led countermeasure designers towards lightweight alternatives. Low-entropy masking schemes [31, 97] reduce the randomness requirements by using masks chosen within a subset of all the possible masks, yet if the leakage function is not linear, they may reduce the security order. Schemes that amortize randomness [85] can achieve similar goals without this shortcoming. In similar lines of work, threshold implementations examined techniques that reduce or even minimize the fresh randomness required to achieve uniformity [34,65]. Still, we stress that several of these schemes need to be evaluated in a fair manner, i.e. by using horizontal leakage exploitation, such as the analysis carried out by Battistello et al. [25] and Grosso et al. [98].

4.1.3 Chapter Organization

This chapter is organized as follows. In Section 4.2 we provide the improved RRM gadgets and analyze the noise amplification stage of RRM. Similarly, we describe RRS and analyze its noise amplification stage in Section 4.3. Conclusions and future directions are discussed in Section 4.4.

4.2 Recycled Randomness Masking - RRM

This section puts forward a low-randomness version of the standard Boolean masking schemes, namely it introduces Recycled Randomness Masking (RRM). The novelty of RRM lies in considering two or more masked multiplications simultaneously and sharing randomness between their compression layers. Using this approach, we develop $t-$NI gadgets that reduce the RNG overhead of masked ciphers and enable side-channel protection at a modest budget. We commence with two elementary examples that will be used throughout this section to illustrate the core recycling idea and we introduce additional notation to describe generic RRM schemes (Section 4.2.1). We continue with section 4.2.2 which searches for optimized $t-$NI randomness-recycling gadgets using formal verification techniques and applies them to the AES cipher. Finally, Section 4.2.3 analyzes the noise amplification stage of RRM schemes and demonstrates the impact of recycling in the noisy leakage model.

4.2.1 Recycling Randomness in Masking

We illustrate the application of RRM using two masked ISW multiplications $z = xy$ and $c = ab$. The multiplications are protected by ISW of security order $d = 1$ (Figure 4.1) or ISW of security order $d = 2$ (Figure 4.2). Both examples assume 4 independent families of shares $(x_i)_{0\leq i\leq d}$, $(y_i)_{0\leq i\leq d}$, $(a_i)_{0\leq i\leq d}$ and $(b_i)_{0\leq i\leq d}$ in $GF(2)$. Values $t_0, t_1, t_2, w_0, w_1, w_2$ are random elements in $GF(2)$ that are necessary to maintain

probing security[2].

In Figures 4.1 and 4.2, red-annotated variables are fresh random elements and

$$
\begin{aligned}
z_0 &= x_0y_0 \oplus \mathbf{w_0} \\
z_1 &= x_1y_1 \oplus (\mathbf{w_0} \oplus x_0y_1) \oplus x_1y_0
\end{aligned}
\qquad
\begin{aligned}
c_0 &= a_0b_0 \oplus (\mathbf{t_0} \leftarrow \mathbf{w_0}) \\
c_1 &= a_1b_1 \oplus ((\mathbf{t_0} \leftarrow \mathbf{w_0}) \oplus a_0b_1) \oplus a_1b_0
\end{aligned}
$$

(*multiplication 1*) (*multiplication 2*)

Figure 4.1: RRM scheme applied on two 1st-order secure ISW multiplications, generating random element w_0 in multiplication 1 and recycling it in multiplication 2.

$$
\begin{aligned}
z_0 &= x_0y_0 & \oplus & & \mathbf{w_0} & & \oplus & \mathbf{w_1} \\
z_1 &= (\mathbf{w_0} \oplus x_0y_1) \oplus x_1y_0 & \oplus & & x_1y_1 & & \oplus & \mathbf{w_2} \\
z_2 &= (\mathbf{w_1} \oplus x_0y_2) \oplus x_2y_0 & \oplus & & (\mathbf{w_2} \oplus x_1y_2) \oplus x_2y_1 & & \oplus & x_2y_2
\end{aligned}
$$

(*multiplication 1*)

$$
\begin{aligned}
c_0 &= a_0b_0 & \oplus & & (\mathbf{t_0} \leftarrow \mathbf{w_0}) & & \oplus & (\mathbf{t_1} \leftarrow \mathbf{w_1}) \\
c_1 &= ((\mathbf{t_0} \leftarrow \mathbf{w_0}) \oplus a_0b_1) \oplus a_1b_0 & \oplus & & a_1b_1 & & \oplus & \mathbf{t_2} \\
c_2 &= ((\mathbf{t_1} \leftarrow \mathbf{w_1}) \oplus a_0b_2) \oplus a_2b_0 & \oplus & & (\mathbf{t_2} \oplus a_1b_2) \oplus a_2b_1 & & \oplus & a_2b_2
\end{aligned}
$$

(*multiplication 2*)

Figure 4.2: RRM scheme applied on two 2nd-order secure ISW multiplications, generating random elements w_0 and w_1 in multiplication 1 and recycling them in multiplication 2.

green-annotated variables are recycled random elements. The left-arrow assignment describes the recycling of a random element in a different multiplication. For instance, in the 1st-order secure ISW-based scheme of Figure 4.1, the element $\mathbf{w_0}$ is generated in multiplication 1 and it is subsequently recycled in multiplication 2 ($\mathbf{t_0} \leftarrow \mathbf{w_0}$). Likewise, the 2nd-order secure example of Figure 4.2 showcases two ISW multiplications, which originally require 6 random elements: (w_0, w_1, w_2) and (t_0, t_1, t_2). To tackle the RNG overhead, RRM generates 3 fresh random elements (w_0, w_1, w_2) during multiplication 1 and recycles w_0 in t_0 and w_1 in t_1. Thus, the amount of random elements required in multiplication 2 is reduced from three to a single random element (t_2).

The proposed recycling technique can be generalized to more than two multiplications of any order and to describe such generic RRM schemes we introduce the following notation. We assume a gadget consisting of n dth-order secure masked multiplications, where every masked multiplication requires s random elements to maintain probing security, e.g. multiplication i requires random elements $(r_{i,1}, r_{i,2}, \ldots, r_{i,s})$. We assume

[2]Throughout this chapter we consider elements in $GF(2)$, yet our results and observations remain applicable in larger fields.

all the inputs to the masked multiplications to be independent families of shares, which may require fresh randomness in our implementation. In addition, we assume that at least one out of n masked multiplications will generate fresh randomness. Subsequently we define a *recycle set* $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n\}$ that consists of sets $\mathcal{R}_i$, $1 \leq i \leq n$. Every set $\mathcal{R}_i$ describes all the the fresh or recycled random elements $r_{i,j}$, $1 \leq i \leq n, 1 \leq j \leq s$ that the multiplication i is using to maintain probing security. Figure 4.1 for instance has $\mathcal{R} = \{\{r_{1,1}\}, \{r_{2,1}\}\} = \{\{w_0\}, \{w_0\}\}$, since the single random element w_0 is generated and used in multiplication 1 and it is reused (recycled) in multiplication 2. Similarly, in Figure 4.2, $\mathcal{R} = \{\{r_{1,1}, r_{1,2}, r_{1,3}\}, \{r_{2,1}, r_{2,2}, r_{2,3}\}\} = \{\{w_0, w_1, w_2\}, \{w_0, w_1, t_2\}\}$, since random elements w_0 and w_1 are generated in multiplication 1 and they are recycled in multiplication 2, while element t_2 is generated in multiplication 2. If only a single multiplication generates fresh random elements and all the other multiplications recycle them, then it holds that $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n\}$, where $\mathcal{R}_i = \mathcal{R}_j$ for all $1 \leq i, j \leq n$. Symmetrically, if no randomness gets recycled (a.k.a. standard masking), then it holds that $R_i \cap R_j = \emptyset$ for all $1 \leq i, j \leq n$ and $i \neq j$. To specify the RNG overhead when recycling, we define the *randomness cost* of an RRM gadget with n multiplications as the total amount of fresh random elements generated. E.g. in Figure 4.1 the randomness cost is 1 and in Figure 4.2 the cost is 4, while in general the cost of an RRM scheme with recycle set $\mathcal{R}$ is $|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|$. The cost of standard masking of n multiplications (without recycling randomness) is equal to $n * s$. In addition, we define the *masking recycle factor* f_{rm} of every random element in the RRM scheme as the number of times it has been used in any multiplication. In the example of Figure 4.1, $f_{rm}(w_0) = 4$, since it occurs twice in every multiplication. Similarly, in the example of Figure 4.2, $f_{rm}(w_0) = f_{rm}(w_1) = 4$ and $f_{rm}(w_2) = f_{rm}(t_2) = 2$. It is noteworthy that the recycling of random numbers is similar to the repeated access to shares observed by Battistello et al. [25], where the recycle factor of a share in dth-order secure scheme is shown to be equal to $d + 1$. We will henceforth refer to a dth-order secure masking gadget with n multiplications and recycle set $\mathcal{R}$ as RRM$(d, n, \mathcal{R})$. It is important to stress that RRM necessitates storing and fetching the recycled random elements. Let *gain* $g = n * s - |\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|$ be the reduction in randomness cost achieved by an RRM$(d, n, \mathcal{R})$ scheme. RRM requires g less random elements and at most g extra storage units, depending on how many times the elements are recycled. In addition, RRM requires at most g extra store and fetch instructions when recycling. For example, in Figure 4.2 the gain $g = 2 * 3 - |\{w_0, w_1, w_2\} \cup \{w_0, w_1, t_2\}| = 2$ and it implies 2 extra storage units (w_0 and w_1), 2 extra store instructions and 2 extra fetch instructions.

4.2.2 Efficient RRM Multiplication Gadgets

As demonstrated, the core contribution of RRM is to reduce the randomness cost of n multiplications below the $n*s$ random elements which are required by standard masking. Notably, both ISW and BBP schemes are already reusing random elements during the compression layer of a single multiplication, while maintaining dth-order probing

security[3]. Still, excessive recycling between multiplications can lead to RRM gadgets that are no longer probing-secure. For instance, assume the ISW-based RRM$(2, 2, \mathcal{R})$ gadget of Section 4.2.1, Figure 4.2 with recycle set $\mathcal{R} = \{\{w_0, w_1, w_2\}, \{w_0, w_1, w_2\}\}$, i.e. 3 fresh elements are generated in multiplication 1 and they are all recycled in multiplication 2. Then, the tuple (z_2, c_2) depends on the sensitive values x, y, a and b simultaneously, because $z_2 \oplus c_2 = x_0y_2 \oplus x_2y_0 \oplus x_1y_2 \oplus x_2y_1 \oplus x_2y_2 \oplus a_0b_2 \oplus a_2b_0 \oplus a_1b_2 \oplus a_2b_1 \oplus a_2b_2$. Since there exists such a tuple (z_2, c_2), the particular RRM gadget is not 2nd-order probing-secure.

As a result, this section proposes $t-$NI optimized multiplication gadgets that are capable to recycle a large amount of randomness. Analytically, for an RRM$(d, n, \mathcal{R})$ gadget, we search for recycle sets $\mathcal{R}$ that minimize the randomness cost, while the gadget remains $t-$NI. We focus on small orders ($d = 1, 2, 3$) and two multiplications per gadget ($n = 2$) due to their practical relevance in implementations. To detect potential security flaws, we use the Lisp-based formal verification tool suggested by Coron [62]. The tool generates all possible tuples of intermediate values (with dimension less or equal to d) that stem from the RRM$(d, n, \mathcal{R})$ gadget and verifies the $t-$NI property using circuit transformations. This process is repeated for all recycle sets $\mathcal{R}$ that ensure the correctness of the scheme, rejecting the insecure choices and identifying the optimized recycle set that minimizes the randomness cost.

Algorithm 1: Brute-force search algorithm.

Input: n: number of multiplications
Input: $\mathcal{R}$: recycle sets
Input: d: security order of multiplications
Input: $\mathcal{T}_d$: d-sized tuples

```
for all R do
    Generate T_d for RRM(d, n, R) ;
    for all tuples t ∈ T_d do
        Verify t−NI of t
    end
    if secure ∀t ∈ T_d then
        Compute RandomnessCost(R)
    end
end
```

The performed brute-force search of Algorithm 1 is carried out for both ISW-based and BBP-based schemes[4] and Figures 4.3a until 4.3e demonstrate the optimized $t-$NI gadgets. The randomness and storage requirements of the proposed RRM gadgets are

[3]ISW uses a symmetric compression structure that reuses every random number once, thus requiring half as many numbers as the naive solution. BBP uses a less regular structure which also reuses every fresh number once.

[4]Running the tool on an Intel Core i7-4719HQ @ 2.5GHz requires minutes to verify 1st and 2nd-order secure RRM gadgets and can reach approximately 5 hours for the verification of 3rd-order secure RRM gadgets.

$z_0 = x_0y_0 \oplus \mathbf{r}_1$
$z_1 = x_1y_1 \oplus (\mathbf{r}_1 \oplus x_0y_1) \oplus x_1y_0$

$c_0 = a_0b_0 \oplus \mathbf{r}_1$
$c_1 = a_1b_1 \oplus (\mathbf{r}_1 \oplus a_0b_1) \oplus a_1b_0$

(a) ISW RRM$(1, 2, \{\{r_1\}, \{r_1\}\})$
randomness/storage cost = 1

$z_0 = x_0y_0 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2$
$z_1 = x_1y_1 \oplus (\mathbf{r}_1 \oplus x_0y_1) \oplus x_1y_0 \oplus \mathbf{r}_3$
$z_2 = x_2y_2 \oplus (\mathbf{r}_2 \oplus x_0y_2) \oplus x_2y_0 \oplus (\mathbf{r}_3 \oplus x_1y_2) \oplus x_2y_1$

$c_0 = a_0b_0 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2$
$c_1 = a_1b_1 \oplus (\mathbf{r}_1 \oplus a_0b_1) \oplus a_1b_0 \oplus \mathbf{r}_4$
$c_2 = a_2b_2 \oplus (\mathbf{r}_2 \oplus a_0b_2) \oplus a_2b_0 \oplus (\mathbf{r}_4 \oplus a_1b_2) \oplus a_2b_1$

(b) ISW RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$
randomness cost = 4, storage cost = 2

$z_0 = x_0y_0 \oplus \mathbf{r}_1 \oplus x_0y_2 \oplus x_2y_0$
$z_1 = x_1y_1 \oplus \mathbf{r}_2 \oplus x_0y_1 \oplus x_1y_0$
$z_2 = x_2y_2 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus x_1y_2 \oplus x_2y_1$

$c_0 = a_0b_0 \oplus \mathbf{r}_1 \oplus a_0b_2 \oplus a_2b_0$
$c_1 = a_1b_1 \oplus \mathbf{r}_2 \oplus a_0b_1 \oplus a_1b_0$
$c_2 = a_2b_2 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus a_1b_2 \oplus a_2b_1$

(c) BBP RRM$(2, 2, \{\{r_1, r_2\}, \{r_1, r_2\}\})$
randomness/storage cost = 2, left-to-right evaluation

$z_0 = x_0y_0 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3$
$z_1 = x_1y_1 \oplus (\mathbf{r}_1 \oplus x_0y_1) \oplus x_1y_0 \oplus \mathbf{r}_4 \oplus \mathbf{r}_5$
$z_2 = x_2y_2 \oplus (\mathbf{r}_2 \oplus x_0y_2) \oplus x_2y_0 \oplus (\mathbf{r}_4 \oplus x_1y_2) \oplus x_2y_1 \oplus \mathbf{r}_6$
$z_3 = x_3y_3 \oplus (x_2y_3 \oplus \mathbf{r}_6) \oplus x_3y_2 \oplus (x_1y_3 \oplus \mathbf{r}_5) \oplus x_3y_1 \oplus (x_0y_3 \oplus \mathbf{r}_3) \oplus x_3y_0$

$c_0 = a_0b_0 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3$
$c_1 = a_1b_1 \oplus (\mathbf{r}_1 \oplus a_0b_1) \oplus a_1b_0 \oplus \mathbf{r}_7 \oplus \mathbf{r}_8$
$c_2 = a_2b_2 \oplus (\mathbf{r}_2 \oplus a_0b_2) \oplus a_2b_0 \oplus (\mathbf{r}_7 \oplus a_1b_2) \oplus a_2b_1 \oplus \mathbf{r}_6$
$c_3 = a_3b_3 \oplus (a_2b_3 \oplus \mathbf{r}_6) \oplus a_3b_2 \oplus (a_1b_3 \oplus \mathbf{r}_8) \oplus a_3b_1 \oplus (a_0b_3 \oplus \mathbf{r}_3) \oplus a_3b_0$

(d) ISW RRM$(3, 2, \{\{r_1, r_2, r_3, r_4, r_5, r_6\}, \{r_1, r_2, r_3, r_7, r_8, r_6\}\})$, randomness cost = 8, storage cost = 4

$z_0 = x_0y_0 \oplus \mathbf{r}_1 \oplus x_0y_3 \oplus x_3y_0 \oplus \mathbf{r}_2 \oplus x_0y_2 \oplus x_2y_0$
$z_1 = x_1y_1 \oplus \mathbf{r}_3 \oplus x_1y_3 \oplus x_3y_1 \oplus \mathbf{r}_2 \oplus x_1y_2 \oplus x_2y_1$
$z_2 = x_2y_2 \oplus \mathbf{r}_4 \oplus x_2y_3 \oplus x_3y_2$
$z_3 = x_3y_3 \oplus \mathbf{r}_4 \oplus \mathbf{r}_3 \oplus \mathbf{r}_1 \oplus x_0y_1 \oplus x_1y_0$

$c_0 = a_0b_0 \oplus \mathbf{r}_1 \oplus a_0b_3 \oplus a_3b_0 \oplus \mathbf{r}_5 \oplus a_0b_2 \oplus a_2b_0$
$c_1 = a_1b_1 \oplus \mathbf{r}_3 \oplus a_1b_3 \oplus a_3b_1 \oplus \mathbf{r}_5 \oplus a_1b_2 \oplus a_2b_1$
$c_2 = a_2b_2 \oplus \mathbf{r}_6 \oplus a_2b_3 \oplus a_3b_2$
$c_3 = a_3b_3 \oplus \mathbf{r}_6 \oplus \mathbf{r}_3 \oplus \mathbf{r}_1 \oplus a_0b_1 \oplus a_1b_0$

(e) BBP RRM$(3, 2, \{\{r_1, r_2, r_3, r_4\}, \{r_1, r_5, r_3, r_6\}\})$, randomness cost = 6, storage cost = 2, left-to-right evaluation

Figure 4.3: Efficient RRM gadgets

demonstrated in Tables 4.1 and 4.2, which confirm that RRM is capable of reducing the randomness cost substantially when compared to standard masking. It remains an open research question to quantify how much the lack of composability (strong non-interference) affects the efficiency, i.e. how many additional refresh layers will be required in the scheme in order to provide a fair comparison with the work of Faust et al. [85].

		Scheme compression					
		ISW security order d			**BBP security order d**		
		1	2	3	1	2	3
Recycling	yes	1	4	8	1	2	6
	no	2	6	12	2	4	8

Table 4.1: Randomness cost of optimized RRM schemes for $n = 2$ multiplications.

		Scheme compression					
		ISW security order d			**BBP security order d**		
		1	2	3	1	2	3
Recycling	yes	1	2	4	1	2	2
	no	0	0	0	0	0	0

Table 4.2: Storage cost of optimized RRM schemes for $n = 2$ multiplications, assuming no storage is needed when recycling within a single multiplication (large register file).

On the application of RRM gadgets to the AES Sbox. Applying these novel randomness-recycling gadgets in the AES cipher is extremely straightforward. Assume a 1st-order secure masked AES cipher that uses the Boyar-Peralta decomposition [40] in the Sbox implementation, i.e. the Sbox requires 32 multiplications in $GF(2)$. During the first execution of the AES cipher, we generate all the necessary random elements without any recycling, i.e. for the first full Sbox execution we need $16 * 32 * 1 = 512$ random elements, resulting in $10 * 512 = 5120$ random elements for 10 rounds of Sbox executions. During the second independent execution of the AES cipher every Sbox multiplication can recycle the randomness generated in the respective multiplication of the first execution, since the gadget RRM$(1, 2, \{\{r_1\}, \{r_1\}\})$ is $t-$NI (Figure 4.3a). Thus, the Sbox-related RNG cost of two AES executions is reduced from 10240 to 5120, i.e. RRM achieves a 50% reduction of the RNG overhead, at the penalty of 5120 element storage, 5120 store and 5120 fetch instructions[5]. In a similar fashion, we can apply the 3rd-order secure BBP-based RRM scheme. During the first AES execution we generate $10 * 16 * 32 * 4 = 20480$ random elements and in the second AES execution we recycle part of them using the specification of the BBP$(2, 2, \{\{r_1, r_2, r_3, r_4\}, \{r_1, r_5, r_3, r_6\}\})$

[5]The actual number of instructions depends on the architecture.

gadget of Figure 4.3e. In this case, RRM achieves a 25% RNG reduction, while storing and fetching 20480 elements. Proving the $t-$NI property for RRM gadgets with more than 2 multiplications ($n > 2$) can enable recycling between more than 2 independent AES executions.

4.2.3 RRM Noise Amplification

The previous section (Section 4.2.2) pinpointed the first pitfall of RRM schemes, i.e. how excessive recycling can result in gadgets that are not probing-secure. Having tackled this issue for low-order gadgets with formal methods, we proceed towards the second pitfall of RRM. Namely, excessive recycling is hazardous to the noise amplification stage of masking, even when the gadget is probing-secure. Specifically, this section analyzes the noise amplification stage of several $t-$NI RRM gadgets of Section 4.2.2, using the mutual information metric suggested by Standaert et al. [197]. In other words, we evaluate the proposed "recycling" countermeasure in the noisy leakage model and compare it to standard masking schemes. The effectiveness of the noise amplification stage of RRM largely depends on the adversary's capability to observe multiple noisy intermediate values during the gadget's execution. We refer to this capability as *horizontal exploitation* and we consider the following cases (C1-C3), in ascending order of adversarial strength:

C1 **Naive-tuple attack.** The adversary exploits a single noisy $(d+1)$-tuple of the RRM gadget and *disregards any repetition* of noisy intermediate values. This scenario is equivalent to an attack against a non-recycling scheme that disregards intra-multiplication repetitions.

C2 **Chosen-tuple attack.** First, the adversary observes the noisy leakage of share repetitions (noted also by Battistello et al. [25]) and the noisy leakage of random element repetitions (noted in this chapter as "randomness recycling") in the gadget. Second, he averages the observed noisy leakages in order to denoise the side-channel emission. Finally, he exploits a *chosen* leakage $(d+1)$-tuple of the RRM gadget that takes advantage of the denoising.

C3 **Full-state attack.** First, the adversary observes the noisy leakage of share repetitions and random element repetitions in the gadget. Second, he averages the observed leakages in order to denoise the side-channel emission. Finally, he exploits the *full state*, i.e. all leaky intermediate values of the RRM gadget.

For our information-theoretic analysis (cases C1-C3), we introduce the following notation to describe the leaky intermediate values and the noisy leakage of RRM gadgets. In a given $(d+1)$-tuple of intermediate values, let random variable S be the sensitive (key-dependent) intermediate value under attack and let random variables $M_0, \ldots, M_{d-1}$ be the masks used to protect the sensitive value. The leakage of a $(d+1)$-tuple is described using the following random vector: $\mathbf{L} = (L_{S \oplus_{i=0}^{d-1} M_i}, L_{M_0}, \ldots, L_{M_{d-1}}) + \mathbf{N}$, where $L_{S \oplus_{i=0}^{d-1} M_i} = \mathsf{L_{id}}\ (S \oplus M_0 \oplus \cdots \oplus M_d)$, $L_{M_i} = \mathsf{L_{id}}\ (M_i)$, $0 \leq i \leq d-1$ and $\mathbf{N}$ is a $(d+1)$-dimensional random vector representing Gaussian noise. We assume

independent and equal noise σ^2 in every sample, i.e. diagonal noise covariance matrix and $L_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $0 \leq i \leq d$.

In the naive-tuple case C1, the adversary disregards the multiple accesses to the family shares (due to the structure of the masking scheme) and also disregards the random element repetition (due to recycling), thus he cannot observe any repeated leakages. In other words, the noise amplification stage in the C1 case is equivalent to that of standard Boolean masking. This naive case is only applicable if the evaluator cannot identify and locate the sample positions of the repeated leakages.

Contrary to C1, the chosen-tuple case C2 assumes that the adversary can locate the leakage sample position of repeated shares and recycled random elements, yet he is still limited to exploit a single $(d+1)$-tuple of leaky intermediate values for his attack. The number of repetitions of a specific random element or share v in the RRM gadget is equal to its recycle factor $f_{rm}(v)$. Averaging all available samples that leak value v results in substantial noise reduction, i.e. $\overline{L_v} \sim \mathcal{N}(\mu_v, \sigma^2/f_{rm}(v))$, which the adversary can use in order to diminish the noise amplification effect of masking. Specifically, he can target a carefully chosen $(d+1)$-tuple of leaky intermediate values, whose leakages have been noise-reduced beforehand. For instance, going back to the example of Section 4.2.1 - Figure 4.2, a sufficient (yet not efficient) attack tuple for RRM(2,2,$\{\{w_0, w_1, w_2\}, \{w_0, w_1, t_2\}\}$) is (x_0y_0, x_1y_1, x_2y_2). Since all the intermediate values of the tuple appear only once, it holds that $L_{x_iy_i} \sim \mathcal{N}(\mu_{x_iy_i}, \sigma^2)$ for $0 \leq i \leq 2$ and the noise amplification is the same as standard masking. A more efficient choice is tuple (z_2, w_1, w_2), where $L_{z_2} \sim \mathcal{N}(\mu_{z_2}, \sigma^2)$, yet $\overline{L_{w_1}} \sim \mathcal{N}(\mu_{w_1}, \sigma^2/4)$ and $\overline{L_{w_2}} \sim \mathcal{N}(\mu_{w_2}, \sigma^2/2)$, because $f_{rm}(w_1) = 4$ and $f_{rm}(w_2) = 2$.

To highlight the effects of recycling on the noisy leakage model, we performed an MI-based evaluation for 1st and 2nd-order secure ISW RRM gadgets that are proposed in Section 4.2.2. We make various choices w.r.t. the recycling factor (f_{rm} ranges from 1 to 10) and the strength of horizontal exploitation (we consider both naive-tuple C1 and chosen-tuple C2 adversaries). The experiments are described in Table 4.3. Naturally, the evaluation depends on the aforementioned parameters, yet we stress that it is adaptable to all RRM choices made by the countermeasure designer. Concretely, computing the MI-metric for a $(d+1)$-tuple requires summing over the randomness vector $\mathbf{M} = (M_0, \ldots, M_{d-1})$ and computing $(d+1)$-dimensional integrations [98]. The resulting MI vs. noise variance plot is visible in Figure 4.4 (left). In addition to the MI-metric, we use the conjecture of Duc et al. [76], in order to approximate the number of traces required to perform a key recovery in the high-noise regime. Analytically, we use the bound $\#traces \geq \frac{H[S]}{MI(S;L)^{d+1}}$ and the no. of traces vs. noise variance plot is visible in Figure 4.4 (right).

$$MI(S;\mathbf{L}) = H[S] + \sum_{s \in \mathcal{S}} Pr[s] \cdot \sum_{\mathbf{m} \in \mathcal{M}^d} Pr[\mathbf{m}] \cdot \int_{\mathbf{l} \in \mathcal{L}^{(d+1)}} Pr[\mathbf{l}|s, \mathbf{m}] \cdot log_2 Pr[s|\mathbf{l}] \;\; d\mathbf{l}$$

$$\text{where } Pr[s|\mathbf{l}] = \frac{\sum_{\mathbf{m}^* \in \mathcal{R}} Pr[\mathbf{l}|s, \mathbf{m}^*]}{\sum_{s^* \in \mathcal{S}} \sum_{\mathbf{m}^* \in \mathcal{R}} Pr[\mathbf{l}|s^*, \mathbf{m}^*]}$$

The evaluation results of 1st and 2nd-order secure RRM (cases C1 and C2) are

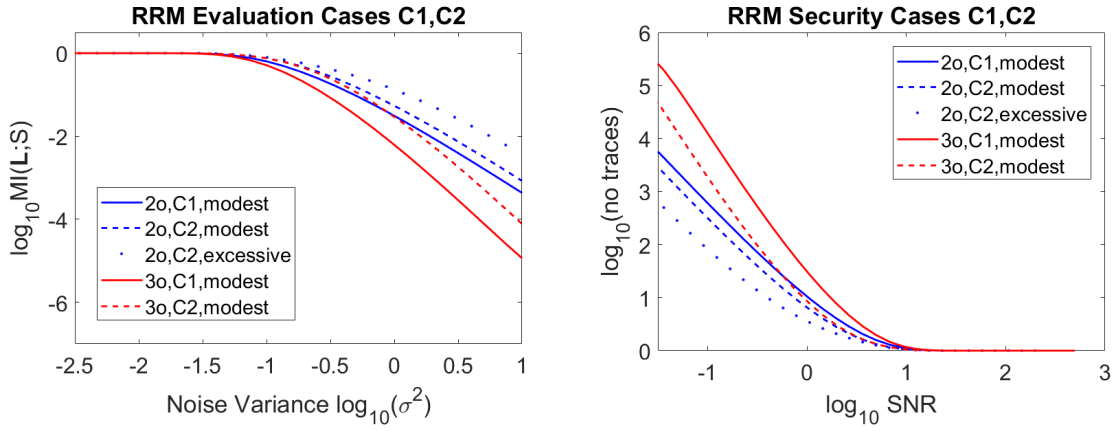


Figure 4.4: MI evaluation and no. traces bound for 1st and 2nd-order secure RRM schemes with 2 multiplications, assuming naive-tuple (C1 - equivalent to std. masking) and chosen-tuple (C2) adversaries. The evaluation considers gadgets with modest and excessive recycling. Blue lines denote 2nd-order attack (vs. 1st-order RRM) and red lines denote 3rd-order attack (vs. 2nd-order RRM).

visible in Figure 4.4 (left), from which we derive three core observations. First, we note that the intermediate values used by the attacker affect directly the RRM evaluation, i.e. the attacker can reduce the security level only by including the average leakage of the repeated random elements or shares in his attack. If the noise-reduced leakages are disregarded (Figure 4.4 solid red and solid blue lines), then the noise amplification remains intact and equivalent to standard masking of the same order. Second, assuming the right tuple is chosen, we observe that increasing the total recycle factor shifts the MI-curve to the right, i.e. the amount of recycling (modest or excessive) indeed damages the noise amplification stage of the scheme. This shift is visible between the dashed blue line (modest recycling) and the dotted blue line (excessive recycling). Note also that excessive recycling may increase the MI of a 2nd-order secure gadget above the MI of a 1st-order secure scheme. Third, we conclude that the RRM technique is in fact a tradeoff between the mutual information level achieved and the randomness cost required. This fact solidifies it as a lightweight alternative to standard masking that can be used by countermeasure designers when the randomness cost becomes prohibitive in a certain application context. Naturally, the designer needs to always be aware of the device's noise level in order to adapt RRM order and recycle set accordingly.

In case C3, the adversary is capable of exploiting the full state of a multiplication, which implies a computational overhead in the MI-formula due to the increase in the integral dimension and the scheme order. In order to bypass this limitation, we simplify the computation of MI, using the approach established by Grosso et al. [98]. Analytically, in order to include the $(d+1)^2$ partial products in our evaluation, we use the information bound established by Prouff et al. [170], stating that the multiplication's leakage is roughly $1.72(d+1)+2.72$ times the leakage of a $(d+1)$-tuple. Computing the bound reduces the evaluation of an RRM multiplication to the evaluation of a

Attack description	**RRM$(d, n, \mathcal{R})$ gadget**	**Attack tuple**	**Recycle factor $\mathbf{f_{rm}}$**
Naive-tuple 2nd-order attack modest recycling	1st-order secure RRM$(1, 2, \{\{r_1\}, \{r_1\}\})$	(x_0y_0, x_1y_1)	$f_{rm}(x_0y_0) = 1$ $f_{rm}(x_1y_1) = 1$
Chosen-tuple 2nd-order attack modest recycling	1st-order secure RRM$(1, 2, \{\{r_1\}, \{r1\}\})$	(z_1, r_1)	$f_{rm}(z_1) = 1$ $f_{rm}(r_1) = 2$
Chosen-tuple 2nd-order attack excessive recycling	1st-order secure RRM$(1, 10, \{\{r_1\}, \ldots, \{r_1\}\})$	(z_1, r_1)	$f_{rm}(z_1) = 1$ $f_{rm}(r_1) = 10$
Naive-tuple 3rd-order attack modest recycling	2nd-order secure RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$	(z_0, z_1, z_2)	$f_{rm}(z_0) = 1$ $f_{rm}(z_1) = 1$ $f_{rm}(z_2) = 2$
Chosen-tuple 3rd-order attack modest recycling	2nd-order secure RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$	(z_2, r_2, r_3)	$f_{rm}(z_2) = 1$ $f_{rm}(r_2) = 4$ $f_{rm}(r_3) = 2$

Table 4.3: $t-$NI RRM gadgets analyzed in the noisy leakage model assuming naive-tuple (C1) and chosen-tuple (C2) adversaries. The attacks exploit a large amount of the available recycling (case C2, excessive recycling) or a small amount of recycling (case C2, modest recycling) or they disregard recycling (case C1).

single $(d+1)$-tuple. We also employ the independent shares' leakage assumption to reduce the leakage vector from the information of a $(d+1)$-tuple $\mathbf{X}$ to the information of a single share X_i, i.e. $\mathbf{L_X} = L_{X_i}$ [76]. However, simplifying to a single-share evaluation does not directly capture the noise reduction issue of RRM, caused by random element and/or share repetitions. To incorporate the noise reduction in our evaluation, we consider the worst-case scenario were the adversary is able to reduce the noise of *all intermediate values* by a recycle factor f_{rm}^{max}. The factor f_{rm}^{max} is the maximum recycle factor observed in any random number or share, e.g. in the gadget RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$, the maximum recycling is observed on random numbers r_1 and r_2, thus $f_{rm}^{max} = 4$. Note that f_{rm}^{max} may stem from either repetitions of random numbers or repetitions of shares. The bound constructed is conservative, since we assume an adversary that can average *every noisy intermediate value* of the encoding by the maximum recycle factor, i.e. $L_{X_i} \sim \mathcal{N}(\mu_{X_i}, \sigma^2/f_{rm}^{max})$, $0 \leq i \leq d$. Still, it provides an efficient alternative to direct computation of the MI formula and demonstrates the evaluation trend for RRM schemes in the high-noise regime. It remains open whether closer bounds can be derived for such scenarios. In Figure 4.5 (left) we demonstrate the MI evaluation of 2nd and 3rd-order secure RRM schemes, with known recycle factor f_{rm}^{max} shown in Table 4.2, using the conservative bound which raises $MI(X_i; L_{X_i})$ to the security order. In Figure 4.5 (right) we demonstrate the no. of traces bound.

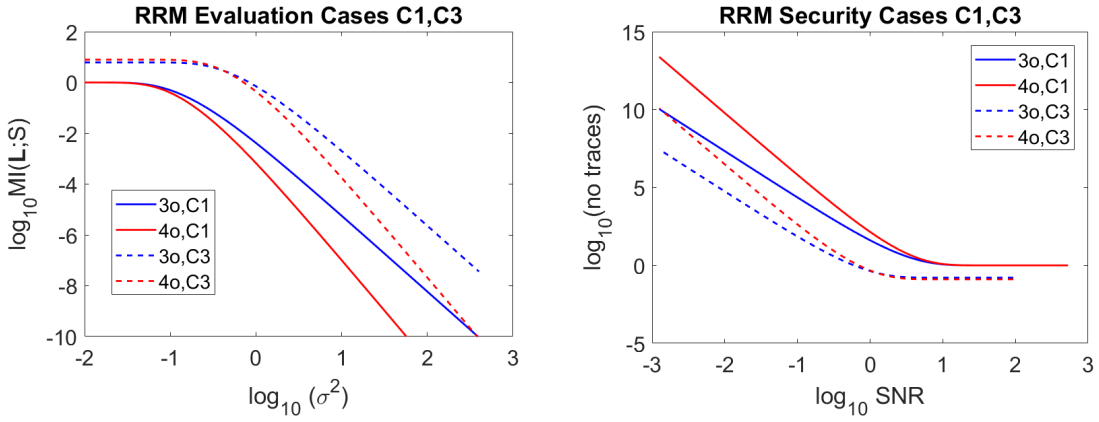


Figure 4.5: MI evaluation for 2nd and 3rd-order secure RRM schemes comparing naive-tuple (C1) with full-state attack (C3). Blue lines denote 3rd-order attack (vs. 2nd-order RRM) and red lines denote 4th-order attack (vs. 3rd-order RMM).

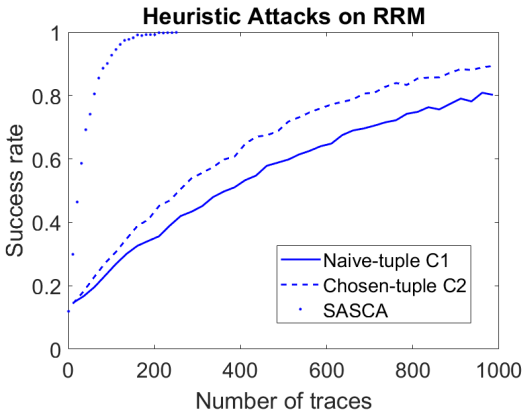
Attack description	**RRM$(d, n, \mathcal{R})$ gadget**	**Attack tuple**	**Recycle factor $\mathbf{f_{rm}}$**
Naive-tuple 3rd-order attack	2nd-order secure RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$	full state	$f_{rm}^{max} = 4$
Chosen-tuple 3rd-order attack	2nd-order secure RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$	full state	$f_{rm}^{max} = 4$
Naive-tuple 4th-order attack	3rd-order secure RRM$(3, 2, \{\{r_1, r_2, r_3, r_4, r_5, r_6\}, \{r_1, r_2, r_3, r_7, r_8, r_6\}\})$	full state	$f_{rm}^{max} = 4$
Naive-tuple 4th-order attack	3rd-order secure RRM$(3, 2, \{\{r_1, r_2, r_3, r_4, r_5, r_6\}, \{r_1, r_2, r_3, r_7, r_8, r_6\}\})$	full state	$f_{rm}^{max} = 4$

Table 4.4: $t-$NI RRM gadgets analyzed using a conservative bound assuming naive-tuple (C1) and full-state (C3) adversaries.

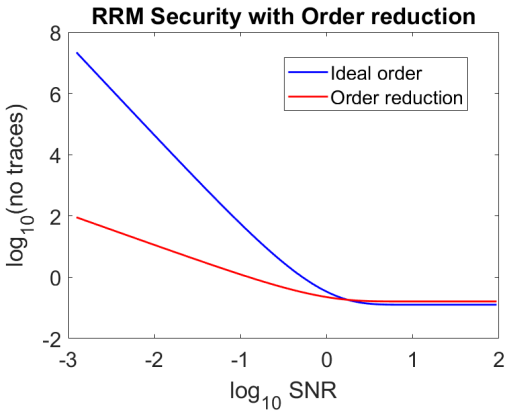
On practical attacks and realistic leakage models. The non-trivial data complexity of the optimal attacks for large order d and large amount of integral dimensions, has led to the development of heuristic attacks that combine the horizontal information of several leaking instructions in a sub-optimal, yet efficient manner. To demonstrate this, we provide two types of *heuristic horizontal attacks* on simulated leakages of a 1st-order secure gadget with 16 multiplications, namely RRM$(1, 16, \{\{r_1\}, \ldots, \{r_1\}\})$. First, we employ the chosen-tuple attack (C2), where the attacker chooses a $(d+1)$-tuple of leakages whose values have been sufficiently denoised by averaging the respective repetitions. The horizontal exploitation of this heuristic attack implies a small overhead for the adversary, namely he needs to perform an averaging pre-processing step. Consecutively, the adversary will employ the noise-reduced tuple in order to attack using Correlation Power Analysis (CPA) [41]. The second heuristic attack that we use in order to exploit horizontally the simulated traceset is a Soft Analytical Side-Channel Attack (SASCA) [215], applied in the context of masking [98]. The SASCA performs the same averaging during the preprocessing step of the first heuristic attack.

Continuing, it exploits the full state of a multiplication by constructing a factor graph and using a belief propagation algorithm. The horizontal exploitation of SASCA implies an overhead depending on the factor graph. The results of the two heuristic attacks and the results of the naive-tuple CPA attack without noise averaging (C1) are visible in Figure 4.6a. As expected, the additional effort w.r.t. horizontal exploitation of the SASCA attack improves the success rate compared to C1 and C2.

Moreover, throughout this chapter we assumed an idealized leakage noisy leakage model, namely independent shares that leak according to the identity function. In practice, several devices showcase order reduction due to various device effects such as glitches, distance-based leakages and coupling [161]. We demonstrate this effect on Figure 4.6b, using a 3rd-order secure RRM scheme and the order-reduction theorem of Balasch et al. [12], which states that distance-based leakages can reduce the security order from d to $\lfloor \frac{d-1}{2} \rfloor$. The red and blue lines of Figure 4.6b give the lower and upper security bounds caused by a large class of real-world leakage flaws.



(a) Success rate of naive, chosen-tuple and SASCA attacks on simulated traces of 1st-order secure RRM with $f_{rm} = 16$.



(b) Security of 3rd-order secure RRM scheme under ideal (blue line) and order-reduced leakage (red line). The order-reduced line is equivalent to 1st-order secure RRM.

Figure 4.6: Practical attacks and realistic leakage models

On the necessity of a noise-based analysis. We conclude this section by showcasing the importance of a noise-oriented analysis of RRM using the following custom scenario. Assume an RRM gadget that recycles a single random number between n 1st-order secure ISW multiplications with mutually independent inputs, where n is large. Trivially, an ISW-based proof shows such a case to be secure, because the adversary can probe only a single intermediate value and thus cannot view multiple recyclings. Thus, we can recycle a random number infinitely while the scheme remains probing-secure. In practice however, the noise level of the leaking random number can be eliminated by averaging the recyclings. A noise-based analysis such as the MI metric or the SASCA can exploit recycling horizontally and is essential to quantify the security damage. If instead the attack remains naive, it may lure the evaluator into a false sense of security.

4.3 Reduced Randomness Shuffling - RRS

Similar to masking, applying the shuffling countermeasure implies a non-negligible randomness cost. Specifically, generating a permutation for shuffling k independent operations of the same type requires $k * \lceil log_2(k) \rceil$ random bits, using a slightly-biased version of the Knuth shuffle algorithm [122, 216]. In a practical scenario, shuffling only 16 AES Sboxes requires 640 random bits in total[6]. In order to deal with this RNG overhead, previous work on the shuffling countermeasure opted to reduce the amount of possible permutations (random start index), to shuffle only in selected rounds (partial shuffling) or to use non-homogeneous shuffle patterns, where the amount of possible permutations varied between cipher layers [103, 181].

Motivated by the recycling ideas of Section 4.2, we use a similar approach on the popular shuffling countermeasure against side-channel analysis. Analytically, we put forward the Reduced Randomness Shuffling (RRS) countermeasure which consists of three shuffling variants that can alleviate the randomness cost involved. In Section 4.3.1 we analyze how RRS reduces the randomness cost compared to standard shuffling. Section 4.3.2 analyzes the susceptibility of RRS to horizontal/multivariate attacks in the noisy leakage model.

4.3.1 Reducing Randomness in Shuffling

To achieve the goal of RNG reduction, we explore the following three variants: *partitioned, merged* and *recycled* shuffling. We demonstrate these three variants using a generic structure of *layers* and *independent operations*. In particular, we assume that the cipher we want to shuffle can be described by the *layer set* $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n\}$ that consists of sets $\mathcal{L}_i$, $1 \leq i \leq n$. Every set $\mathcal{L}_i$ describes s independent operations that constitute this layer, e.g. $\mathcal{L}_i = \{o_{i,1}, o_{i,2}, \ldots, o_{i,s}\}$. The partitioning of a cipher into layers and of layers into independent operations rests upon the countermeasure designer and it is closely related to the cipher implementation. For instance, the independent operations may range from whole cipher parts (e.g. shuffling Sboxes) to individual assembly operations (e.g. shuffling key-dependent instructions). We will refer to a Reduced Randomness Shuffling scheme that shuffles independent operations according to the layer set $\mathcal{L}$ as RRS($\mathcal{L}$). In addition we specify the *randomness cost* of the RRS countermeasure as the total RNG overhead required to shuffle the cipher according to layer set $\mathcal{L}$. Figures 4.7a-4.7d illustrate the application of RRS on a layered structure.

The example of Figure 4.7a commences with an RRS scheme that shuffles a cipher structure, using $n = 2$ layers and $s = 4$ independent operations per layer, i.e. layer set $\mathcal{L} = \{\{o_{1,1}, o_{1,2}, o_{1,3}, o_{1,4}\}, \{o_{2,1}, o_{2,2}, o_{2,3}, o_{2,4}\}\}$. Both layers are shuffled using two different permutations on 4 operations, namely permutations $\mathbf{P}_4^{\mathcal{L}_1}$ and $\mathbf{P}_4^{\mathcal{L}_2}$. Thus, the randomness cost for a single execution of the 2-layer structure is $|\mathcal{L}| * |\mathcal{L}_i| * \lceil log_2(|\mathcal{L}_i|) \rceil = 2 * 4 * log_2(4) = 16$ bits.

[6]The cipher runs for 10 rounds, permuting 16 independent operations of the same type (Sbox) per round. Every permutation requires $16 * \lceil log_2(16) \rceil$ random bits.

To scale down the randomness cost, *partitioned shuffling* splits *vertically* a set of independent operations $\mathcal{L}_i$ into two or more smaller subsets that are cheaper to shuffle. For instance, in Figure 4.7b, instead of shuffling a single set of 4 independent operations $\mathcal{L}_1 = \{o_{1,1}, o_{1,2}, o_{1,3}, o_{1,4}\}$, we opt to partition $\mathcal{L}_1$ in two subsets of 2 independent operations each. Thus, we will partition $\mathcal{L}_1$ to $\mathcal{L}'_1 = \{o_{1,1}, o_{1,2}\}$ and $\mathcal{L}''_1 = \{o_{1,3}, o_{1,4}\}$. An analogous partitioning is done in $\mathcal{L}_2$, resulting in $\mathcal{L}'_2 = \{o_{2,1}, o_{2,2}\}$ and $\mathcal{L}''_2 = \{o_{2,3}, o_{2,4}\}$. We define the granularity of this vertical partitioning as the *partition factor* f_p, where $f_p = 1$ implies no partitioning. Performing partitioned shuffling with factor f_p on $|\mathcal{L}_i|$ independent operations reduces the randomness cost of layer i to $|\mathcal{L}_i| * \lceil log_2(|\mathcal{L}_i|/f_p)\rceil$. In the example of Figure 4.7b, we use $f_p = 2$ on both cipher layers and we replace $\mathbf{P}_4^{\mathcal{L}_1}$ and $\mathbf{P}_4^{\mathcal{L}_2}$ with $\mathbf{P}_2^{\mathcal{L}'_1}$ and $\mathbf{P}_2^{\mathcal{L}'_2}$ respectively, reducing the cost of a single execution from 16 to 8 bits.

To similar ends, the *merged shuffling* variant combines several cipher layers *horizontally* in order to permute them together. The example of Figure 4.7c views $\mathcal{L}_1$ and $\mathcal{L}_2$ as a single layer and shuffles them using the same permutation. That is, we merge $\mathbf{P}_4^{\mathcal{L}_1}$ and $\mathbf{P}_4^{\mathcal{L}_2}$ into permutation $\mathbf{P}_4^{\mathcal{L}''}$, s.t. $\mathcal{L}'' = \{\{o_{1,1}, o_{2,1}\}, \{o_{1,2}, o_{2,2}\}, \{o_{1,3}, o_{2,3}\}, \{o_{1,4}, o_{2,4}\}\}$. We define the granularity of this horizontal combination as the *merge factor* f_m, where $f_m = 1$ implies no merging and observe that merged shuffling can reduce the randomness cost of a single iteration to $(|\mathcal{L}|/f_m) * k * \lceil log_2(|\mathcal{L}_i|)\rceil$. Naturally, merging and partitioning can be combined, resulting in randomness cost of $(|\mathcal{L}|/f_m) * |\mathcal{L}_i| * \lceil log_2(|\mathcal{L}_i|/f_p)\rceil$ bits per iteration. Still, different cipher layers can present a different number of independent operations for partitioned/merged shuffling and thus may need to be homogenized by shuffling additional dummy operations.

Last, *recycled shuffling* opts for the "external" recycling of the generated permutation, i.e. we reuse a permutation between different executions or rounds of the cipher structure. In Figure 4.7d, the layer $\mathcal{L}_1$ of cipher execution no. 1 and the layer $\mathcal{L}_1$ of cipher no. 2 are independent, yet they are shuffled with the same permutation $\mathbf{P}_4^{\mathcal{L}_1}$. We define the *recycle factor of shuffling* f_{rs} as the number of repetitions of a permutation in different cipher iterations, i.e. $f_{rs} = 1$ implies no recycling. Recycled shuffling can reduce the randomness cost of a certain layer i from $(\#executions) * |\mathcal{L}_i| * \lceil log_2(|\mathcal{L}_i|)\rceil$ to $(\#executions/f_{rs}) * |\mathcal{L}_i| * \lceil log_2(|\mathcal{L}_i|)\rceil$.

We note that only recycled shuffling implies an overhead due to storage units and store/fetch instructions, while partitioned and merged shuffling simply use less randomness. The overhead relates to the recycle factor of shuffling, i.e. reusing the same permutation results in f_{rs} extra store/fetch instructions and a memory unit to store the random number.

On the application of RRS to the AES cipher. Assume that the countermeasure designer focuses on the first two layers of the AES cipher, namely KeyAddition (ka) and Sbox (sb). The standard way to shuffle them would require two permutations on 16 independent operations, i.e. $\mathbf{P}_{16}^{KA}$ and $\mathbf{P}_{16}^{S}$, costing 128 random bits per round, resulting in 1280 bits for 10 rounds of AES. Alternatively, the designer can opt to partition both layers with partition factor $f_p = 4$, i.e. split $\{ka_1, \ldots, ka_{16}\}$ and $\{sb_1, \ldots, sb_{16}\}$ into sets

$\{ka_1, \ldots, ka_4\}, \{ka_5, \ldots, ka_8\}, \{ka_9, \ldots, ka_{12}\}, \{ka_{13}, \ldots, ka_{16}\}$ and $\{sb_1, \ldots, sb_4\}$, $\{sb_5, \ldots, sb_8\}$, $\{sb_9, \ldots, sb_{12}\}$, $\{sb_{13}, \ldots, sb_{16}\}$ respectively. Thus, the cost is reduced to 640 bits ($\mathbf{P}_4^{KA}$ and $\mathbf{P}_4^{S}$ for 50% RNG reduction). In a similar fashion, the designer can merge the KeyAddition and Sbox layers into a single layer, i.e. $\mathcal{L} = \{kasb_1, \ldots, kasb_{16}\}$, reducing again the cost to 640 bits ($\mathbf{P}_{16}^{KA,S}$ for 50% RNG reduction). Finally, any generated permutations on KeyAddition, Sbox can be recycled in subsequent AES executions, reducing RNG even further, at the penalty of extra storage.

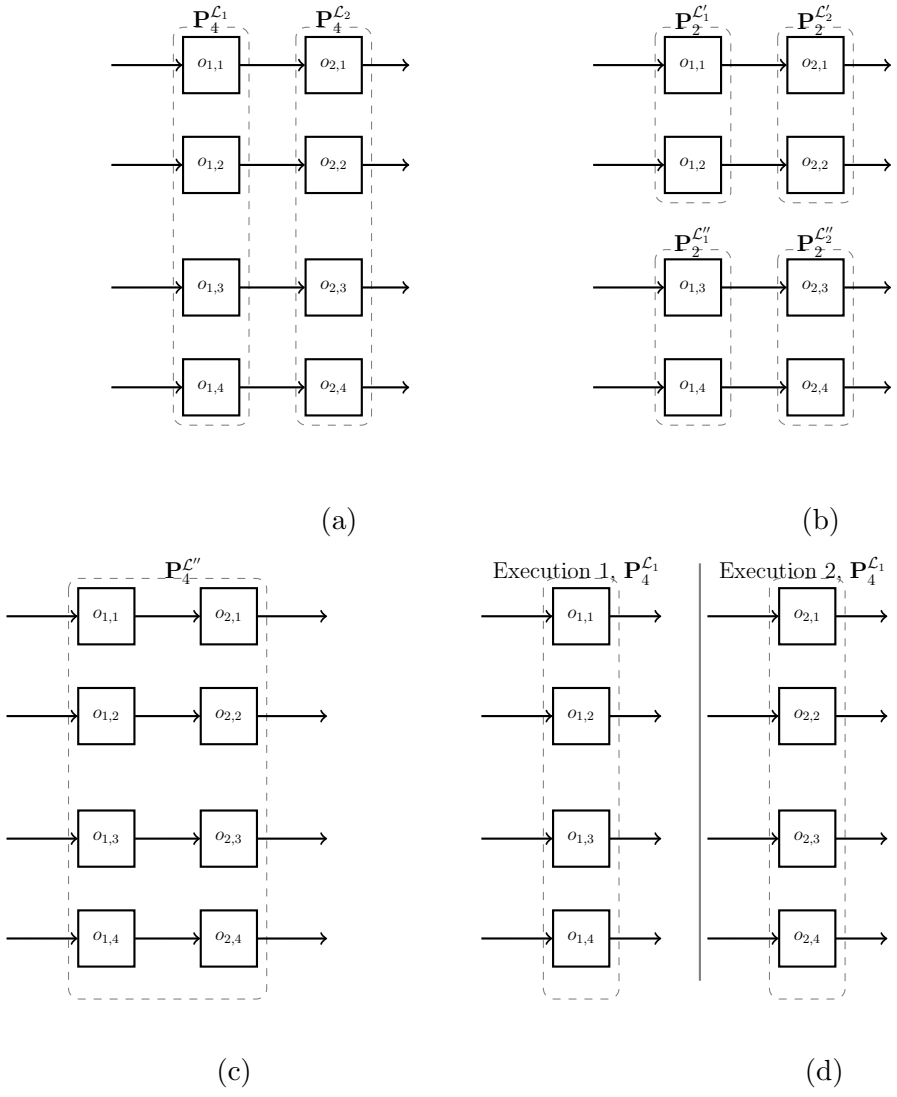


Figure 4.7: Initial, partitioned, merged and recycled shuffle is applied to the layered cipher structure in Figures (a) - (d). Dashed-line boxes indicate the operations and layers that are shuffled with the same permutation. The arrows indicate the information flow between layers.

4.3.2 RRS Noise Amplification

As expected, reducing the randomness cost of shuffling has a direct impact on the noise amplification effect of the countermeasure, offering an interesting randomness-security tradeoff for the designer. Similarly to Section 4.2.3, we evaluate the variants of RRS via the mutual information framework and consider an adversary that can exploit horizontally more than a single cipher layer. We perform our evaluation on the layered cipher structure used previously, where the adversary attempts to recover any part of the key $\mathbf{k} = (k_0, k_1, k_2, k_3)$ that is related to the 4 independent operations of $\mathcal{L}_1$ and $\mathcal{L}_2$. To that end, he may exploit the leakage from both layers as well as direct leakage from the permutations used to shuffle these layers. Below, we introduce the random variable notation that describes shuffling in the noisy leakage model.

- The adversary can observe the leakage vector after every cipher layer, namely $\mathbf{L}^{\mathcal{L}_1}$ and $\mathbf{L}^{\mathcal{L}_2}$. The leakage variables L_i depend on the layer permutations $\mathbf{P}_n^{\mathcal{L}_1}$ and $\mathbf{P}_n^{\mathcal{L}_2}$, thus it holds that $L_i^{\mathcal{L}_1} = L_{id}\left(X_{P_i^{\mathcal{L}_1}}\right) + noise$ and $L_i^{\mathcal{L}_2} = L_{id}\left(Y_{P_i^{\mathcal{L}_2}}\right) + noise$, where $noise$ represents additive Gaussian noise $\mathcal{N}(0, \sigma^2)$.
- The adversary can observe the direct permutation leakage of every shuffled layer, namely $\mathbf{L}'^{\mathcal{L}_1}$ and $\mathbf{L}'^{\mathcal{L}_2}$. For layer permutations $\mathbf{P}_n^{\mathcal{L}_1}$ and $\mathbf{P}_n^{\mathcal{L}_2}$, it holds that $L_i'^{\mathcal{L}_1} = L_{id}\left(P_i^{\mathcal{L}_1}\right) + noise$ and $L_i'^{\mathcal{L}_2} = L_{id}\left(P_i^{\mathcal{L}_2}\right) + noise$, where $noise$ represents additive Gaussian noise $\mathcal{N}(0, \sigma^2)$.

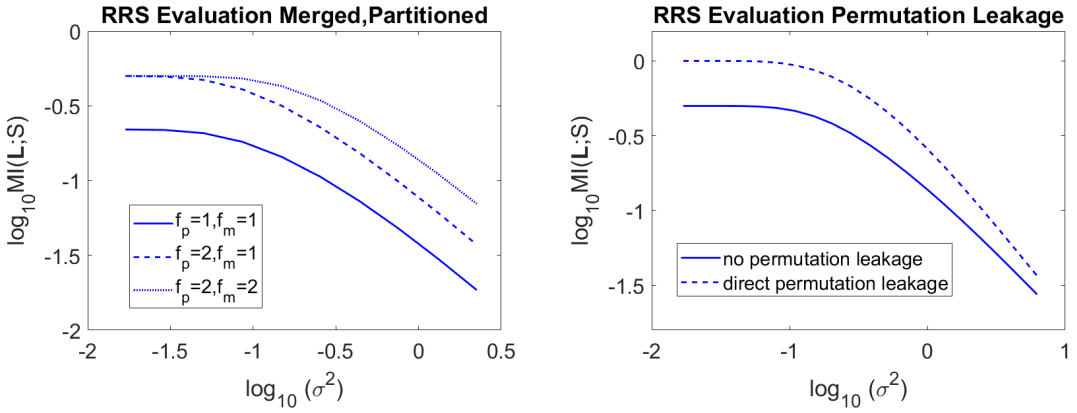
To analyze the tradeoff between the MI level and the randomness cost, we perform the MI-based evaluation for several versions of the RRS and attack options. The cases are demonstrated in Table 4.5. The evaluation uses the formula by Charvillon et al. [216], which we update in order to account for the partitioned, merged and recycled shuffling with factors f_p, f_m and f_{rs} respectively.

$$MI(K_t; \mathbf{L}) = H[K_t] + \sum_{k_t \in \mathcal{K}_t} Pr[k_t] \cdot \int_{\mathbf{l} \in \mathcal{L}^\eta} Pr[\mathbf{l}|k_t] \cdot log_2 Pr[k_t|\mathbf{l}] \; d\mathbf{l} \quad \text{where}$$

$$Pr[k_t|\mathbf{l}] = \frac{Pr[\mathbf{l}|k_t]}{\sum_{k_t^* \in \mathcal{K}_t} Pr[\mathbf{l}|k_t^*]} \;\text{ and }\; Pr[\mathbf{l}|k_t] = \sum_{\mathbf{p} \in \mathcal{P}_\theta} \frac{Pr[\mathbf{l}'|\mathbf{p}]}{\sum_{\mathbf{p}^* \in \mathcal{P}_\theta} Pr[\mathbf{l}'|\mathbf{p}^*]} \cdot Pr[\mathbf{l}|k_t, \mathbf{p}]$$

In the formula above, we assume that the adversary attacks a certain key part K_t, where $t \in \{0, 1, 2, 3\}$. We also note that the adversary in general exploits η-dimensional leakage vectors $\mathbf{L}, \mathbf{L}'$ and performs summations over the set of θ-dimensional permutations. In the following analysis we show how parameters η and θ relate to the particular RRS variant used as well as the adversary's horizontal capabilities (i.e. the number of layers attacked). The results of the MI-based evaluation are visible in Figures 4.8a,4.8b.

In Table 4.5, case D1 , the adversary attacks a standard shuffling scheme where no randomness reduction is performed. We assume again the reduced block cipher structure with two layers, $k = 4$ independent operations per layer and a two 4-dimensional permutations $\mathbf{P}_4^{\mathcal{L}_1}$, $\mathbf{P}_4^{\mathcal{L}_2}$ for shuffling them. The adversary attacks a single



(a) MI evaluation for partitioned and merged RRS schemes on 2 layers, without permutation leakage (cases D1, D2, D3). (b) MI evaluation with and without direct permutation leakage, for parameters $f_p = 2, f_m = 2, frs = 2$ (cases D4, D5).

Figure 4.8: RRS MI evaluation

Case	No. of layers attacked	Direct permutation leakage exploited	Partition factor f_p	Merge factor f_m	Recycle factor f_{rs}
No RRM (D1)	1	no	1	1	1
Partitioned (D2)	1	no	2	1	1
Partitioned, merged (D3)	2	no	2	2	1
Partitioned, merged, recycled (D4)	2	no	2	2	2
Partitioned, merged, recycled, direct perm. leakage (D5)	2	yes	2	2	2

Table 4.5: Reduced Randomness shuffling schemes analyzed in the context of single-layer and two-layer horizontal attacks, with/without direct permutation leakage.

layer, i.e. he focuses solely on the 4-dimensional leakages $\mathbf{L}^{\mathcal{L}_1}$ observed in layer $\mathcal{L}_1$. Disregarding layer $\mathcal{L}_2$ implies integration over 4-dimensional leakages, i.e. parameter $\eta = k = 4$. In addition, the permutation that the adversary needs to consider in the attack is 4-dimensional, resulting in parameter $\theta = k = 4$. Continuing, case D2 evaluates a single-layer attack on an RRS scheme where both $\mathcal{L}_1$ and $\mathcal{L}_2$ layers use partitioned shuffling with factor $f_p = 2$. In this case, the adversary evaluates by focusing on the 2-dimensional leakages of layer $\mathcal{L}_1$, which are shuffled by a 2-dimensional permutation $\mathbf{P}_2^{\mathcal{L}_1}$. In other words, it holds that $\eta = \theta = k/f_p = 2$. The MI curve of D2 is shifted to the right of curve D1, so we show that reducing the number of available permutations via partitioned shuffling is detrimental to the noise amplification stage.

Case D3 evaluates a two-layer attack on an RRS scheme that combines partitioned and merged shuffling with factors $f_p = 2$ and $f_m = 2$. Specifically, layers $\mathcal{L}_1$ and $\mathcal{L}_2$ are merged and shuffled together with permutation $\mathbf{P}_2^{(\mathcal{L}_1,\mathcal{L}_2)}$. The adversary takes advantage of this fact and targets both layers in order to extract more information horizontally. The attack uses leakage vectors $(\mathbf{L}^{\mathcal{L}_1}, \mathbf{L}^{\mathcal{L}_2})$, so $\eta = (\#no\_attacked\_layers) * (k/f_p) = 4$ and the only permutation in place is $\mathbf{P}_2^{(\mathcal{L}_1,\mathcal{L}_2)}$, thus

$\theta = k/f_p = 2$. The MI curve of D3 is shifted to the right of curve D2, showing that merged shuffling can improve the effectiveness of multi-layer horizontal attacks and it is detrimental to the MI level.

Last, we compare partitioned, merged and recycled shuffling (case D4) with equivalent shuffling that can observe the repeated direct permutation leakage. Specifically, in both cases, the adversary exploits horizontally two partitioned layers that use the same permutation, i.e. $\eta = 4$ and $\theta = 2$. Note however, that in case D5 the adversary can also observe the repeated direct permutation leakage, i.e. he has access to $\mathbf{L}'^{\mathcal{L}_1}_j$ for all executions $j = 1, \ldots, f_{rs}$, while D4 assumed equiprobable permutations. As a result, in case D5, the adversary can reduce the noise level of the direct permutation leakage by computing $\overline{\mathbf{L}'^{\mathcal{L}_1}} \sim \mathcal{N}(\boldsymbol{\mu}^{\mathcal{L}_1}, (1/f_{rs}) * \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ a diagonal covariance matrix. Figure 4.8b shows how exploiting the direct permutation leakage enhances the attack.

4.4 Conclusions & Future Directions

In this chapter, we have performed an in-depth investigation of low-randomness alternatives to standard masking and shuffling, namely RRM and RRS. The first core outcome is that RRM and RRS can offer effective tradeoffs between randomness cost and security. A designer of side-channel countermeasures can now rely on the MI-based evaluation and provide optimized and flexible protection that reduces the randomness cost. The second core outcome of this chapter is demonstrating the importance of horizontal exploitation in masking and shuffling. We have shown that univariate (or partially horizontal) evaluations provide us with only a part of the whole picture and may lure the evaluator into a false sense of security. By examining the multivariate adversarial model, we exploit a larger quantity of the available leakage and provide a more complete security evaluation. This notion of multivariate security will be revisited in Chapter 5, where similar horizontal attacks can be deployed. Last, this chapter has demonstrated the necessity of noise-based analysis as a complement to formal methods. We maintain that a sound evaluation approach is to start from a provably secure scheme and enhance it with a noise-based analysis in order to provide a more holistic view.

With regards to future work, we note that multivariate evaluation techniques are still at a nascent stage when it comes to real-world devices. In fact, research efforts concentrate on a fairly high abstraction layer, i.e. they only consider leaky cipher operations. As a result, they disregard many peculiarities and defects of the hardware and physical layers that we have encountered in Chapter 3. Future research needs to strive towards integrating this complexity in our evaluations and improve the attacks that are able to exploit horizontally RRM and RRS. Moreover, we have just begun unveiling the subtle interactions between countermeasures, randomness and security. This chapter marks the first departure from the common triptych of design-implementation-evaluation and tries to reconfigure the countermeasure's security through RNG. The next chapter will keep investigating such interactions and tradeoffs, albeit in the context of fault injection countermeasures.

Chapter 5

The Price of Fault Resistance

"Remember to pay attention to real objects in time and space and not lose them in utterly idealized abstractions. Remember that the qualitative effects of context and interaction may be lost when phenomena are isolated."

Richard Lewontin, 2009

Confirming the quote of Richard Lewontin on dialectical biology, Chapter 3 has demonstrated extensively how idealized abstractions of masking schemes can fall short of our expectations when applied in a real-device context. Continuing, Chapter 4 has highlighted the interaction between randomness and side-channel resistance, two phenomena that were often treated separately. In the same spirit, this chapter studies the interaction between two frequently isolated phenomena, namely side-channel resistance and fault injection resistance. We shall see that they constitute opposing forces in hardware security and thus, similarly to Chapter 4, we yearn for a clear understanding of the subtle interactions between them.

Side-channel analysis and fault injection attacks have coexisted since the early days of hardware security, with early FI attacks dating back to 1997, where Boneh et al. showed that only a single computation fault is sufficient to extract an RSA private key [37]. Like SCA, fault injection has become a rich research field where designers and implementors are working together to secure cryptographic primitives. In this chapter we focus on two core research trends of the FI field and subsequently use them to clarify the interactions between FI and SCA. Analytically, we focus first on popular countermeasures against FI such as duplication, n-plication and infection [210], all of which attempt to thwart faults by adding redundancy to an existing cipher (we shall refer to this as *a posteriori* redundancy). Second, we focus on a more recent trend, namely new ciphers such as FRIET [190], which attempt to tackle the FI problem during the early stages of cipher design. This approach designs a cipher with built-in protection against faults and integrates error detection directly in the cipher structure (we shall refer to this as *a priory* redundancy). Both trends are subsequently examined from the viewpoint of a side-channel adversary, who will exploit these diverse forms of redundancy and enhance his SCA attack.

This chapter is based on work published in [59] and [190] and can be viewed as a continuation of the works of Regazzoni et al. [175]. It is motivated by the need to

study the interactions between SCA and FI and balance the tradeoff between these two opposing forces. Throughout this chapter we will confirm once again the quoted warning, that is if we treat side-channels and fault injection resistance in an isolated manner, the evaluator can be lured into a false sense of security. On the contrary, viewing hardware security as an interaction between SCA and FI can yield a better understanding and provide new options for the countermeasure designer. The main points of the chapter are summarized below.

- This chapter demonstrates the "price" of fault resistance, i.e. it assesses the impact of FI countermeasures (a posteriori redundancy) or FI-resistant ciphers (a priori redundancy) on side-channel analysis resistance. The chapter employs several multivariate exploitation techniques in theory and practice and demonstrates that the added redundancy can enhance the side-channel adversary and result in stronger attacks.

- Since most secure hardware or software elements require protection against both side-channel and fault attacks, this chapter can serve as a guideline for the countermeasure designer. Whether he opts to add redundancy on an existing cipher or he opts for a new cipher with built-in redundancy features, the designer can customize the fault injection resistance, while minimizing its impact on side-channel resistance. Note that this adaptive process can be carried out in conjunction with fine-tuning RNG and side-channel resistance (as shown in Chapter 4), thus it expands the design space of SCA/FI countermeasures.

5.1 Introduction

When developing a secure device, we do not often encounter cases where side-channel analysis is the sole security requirement. Another big class of hardware attacks and exploits relies on the fault injection techniques. The later can change the data under computation, skip critical parts of the control logic and in general alter the behavior of the underlying circuitry. To prevent fault attacks, a multitude of countermeasures/structures have been investigateds. Software-based instruction redundancy methods for fault detection were put forward by Barenghi et al. [16]. In this technique, the original stream of instructions to be executed is duplicated (or even triplicated), one instruction after another, either manually or automatically [18, 137, 151]. To the same end, Tupsamudre et al. [210] put forward the infection countermeasure, where dummy rounds diffuse the injected faults, making attacks harder. Ishai et al. [110] extended the well-known ISW masking scheme to protect against tampering for an arbitrary order of circuit alterations. Naturally, all these algorithmic countermeasures imply additional computation and rely on *redundancy* to achieve their stated goals. However, any form of redundancy during a cipher computation implies a larger amount of side-channel leakage information. Thus, motivated by the increased amount of available information, this chapter investigates the relationship between SCA and FI resistance.

5.1.1 Chapter Contribution

Starting, this chapter takes an in-depth look at the tradeoff between FI resistance and SCA resistance by investigating FI countermeasures and new cipher structures with build-in FI protection.

We examine the interaction between n-plication and side-channel resistance and demonstrate the theoretical tradeoff using an information-theoretic approach. Similarly, we showcase the impact of infection [210] countermeasures on side-channel security by employing the Hidden Markov Model. On the experimental side, we show how practical horizontal exploitation techniques can leverage the extra side-channel information introduced by duplication-based defenses. The results are based on experimental acquisitions on AVR XMEGA128D4 and demonstrate improvements over naive CPA-based techniques.

In addition, this chapter describes the novel FI-resistant design of FRIET [190], a cipher with build-in fault detection capabilities. In order to assess how the build-in FI resistance affects side-channel attacks, we employ the Soft-Analytical Side-Channel Attack to exploit the leakage emitted by the full FRIET round. The results show that fault-detecting structure is exploitable and can indeed enhance SCA. Moreover, we demonstrate that this structure reveals less information compared to duplication, while both techniques provide the same level of FI resistance.

5.1.2 Previous Work

In this section, we describe the existing work that investigated the interactions between FI and SCA. First, we iterate through the previous work that identified the SCA impact of FI countermeasures. Second, we iterate through different lightweight cipher designs and observe how they are becoming increasingly aware of hardware security, including FI attacks.

FI countermeasures. Regarding a posteriori redundancy, Regazzoni et al. [177] first looked at the interaction between fault injection defenses and side-channel attacks. Specifically, they studied an AES implementation with parity-based error detection circuitry. They conclude that the presence of a parity error detection circuit will leak important information to an attacker through side-channels. One year later, Regazzoni et al. [176] experimentally show the exploitability of a known-by-the-attacker error detection circuit. Pahlevanzadeh et al. [156] look at three fault detection methods designed specifically for AES: double module redundancy, parity checks, inverse execution; all implemented on an FPGA. They find that parity checks are actually improving the resistance against standard CPA. Similarly, Luo et al. [136] use CPA to attack an FPGA implementation of AES which is capable of fault detection. They conclude that duplication does not improve the success rate of the attack in respect to the unprotected AES implementation. However, we observe that the approaches of [136, 156] use *naive* CPA attacks and do not rely on multivariate, horizontal exploitation of the leakage. We stress that such techniques can possible ignore parts of the leakage, thus they do not reveal the full picture and may lure the side-channel evaluator in a false sense of security. This chapter works towards amending such approaches.

FI-resistant design. Historically, hardware-oriented cipher design was primarily geared towards cryptography that operates within limited area resources. Thus, the initial focus was proposing block ciphers with small block size and a round function that can be implemented compactly, culminating in the design of PRESENT [36], KATAN/KTANTAN [44] and others. Recently, newer optimization targets emerged. Low-latency led to PRINCE [38], tweakability led to Skinny and Mantis [27] and low-power led to MIDORI [15]. Regarding side-channel resistance as an optimization target, Noekeon [207] and the LS designs [95] factored in their design the number of non-linear components. Thus, they manage to reduce the quadratic performance and RNG overhead caused by masking multiplications. Most of the existing designs do not take into account resistance to fault injection. Instead, FI resistance is considered as an add-on solution that should be dealt with in the actual implementation. In this chapter, we present a cipher that has build-in FI resistance and we work towards a fair side-channel evaluation for this structure.

5.1.3 Chapter Organization

The chapter is organized as follows. Section 5.2 analyzes the price of a posteriori redundancy (FI countermeasures) on side-channel resistance. Likewise, Section 5.3 an-

alyzes the price of a priory redundancy (FI-resistant cipher) on side-channel resistance. Conclusions and future directions are provided in Section 5.4.

5.2 SCA Evaluation of FI Countermeasures

This section demonstrates the interactions between the redundancy-based FI countermeasures and the side-channel resistance of an implementation that is employing them. In Section 5.2.1 we analyze the theoretical impact of instruction duplication and n-plication on SCA using an information-theoretic approach. Section 5.2.2 demonstrates how to perform SCA on infective countermeasures using a Hidden Markov Model that simplifies the exploitation phase of infection and makes it equivalent to instruction duplication.

5.2.1 Theoretical Evaluation of Instruction Duplication

From the side-channel perspective, the ID countermeasure increases the available leakage in a horizontal manner, either as a fault detection or as a fault tolerance mechanism. Analytically, in the case of an unprotected implementation (without ID) a univariate adversary can acquire the leakage of a key-dependent value v, i.e. observe $L_v \sim \mathcal{N}(v, \sigma)$, assuming identity leakage model. On the contrary, when instruction n-plication is implemented ($n > 1$), the adversary can observe over time an n-dimensional leakage vector $\mathbf{L}_v = [L_v^{t=1}, \ldots, L_v^{t=n}]$. The vector contains n independent observations of value v under the same noise level (homoscedastic noise), i.e. we assume that $L_v^t \sim \mathcal{N}(v, \sigma)$, $t = 1, \ldots, n$.

Given that the side-channel adversary has located the sample positions of the repeated leakages, he can perform a pre-processing step where he averages all available samples that leak v, i.e. he computes $\bar{L}_v = (1/n) * \sum_{t=1}^{n} L_v^t$. Exactly like averaging recycled randomness in Chapter 4, this step results in noise reduction of factor $\sqrt{n}$, obtaining $\bar{L}_v \sim \mathcal{N}(v, \sigma/\sqrt{n})$ and as a result side-channel attacks can be enhanced. Note that noise reduction can be particularly hazardous even when additional side-channel protection is implemented. For instance, both masking and shuffling countermeasures amplify the existing noise of a device [201, 216] and will perform poorly if the noise level has been reduced by a large factor $\sqrt{n}$. In order to demonstrate the effect of noise reduction, we employ the information-theoretic framework of Standaert et al. [197] which evaluates the resistance against the worst possible attack scenario. The MI between the key-dependent value V and leakage $\mathbf{L}_v$ can be computed using the following formula. The formula is exactly the same as the standard MI formula, we just stress that value V is an intermediate value that is both key-dependent and n-plicated because of the FI countermeasures. The leakage dimensionality n changes accordingly to the FI countermeasures, e.g. $n = 1$ for unprotected, 2 for duplication, etc. The results, including the averaging step, are visible in Figure 5.1.

$$MI(V;\mathbf{L}_v) = H[V] + \sum_{v\in\mathcal{V}} Pr[v] \cdot \int_{\mathbf{l}_v\in\mathcal{L}^n} Pr[\mathbf{l}_v|v] \cdot \log_2 Pr[v|\mathbf{l}_v]\ \mathrm{d}\mathbf{l}_v,$$
$$\text{where } Pr[v|\mathbf{l}_v] = \frac{Pr[\mathbf{l}_v|v]}{\sum_{v^*\in\mathcal{V}} Pr[\mathbf{l}_v|v^*]} \tag{5.1}$$

.

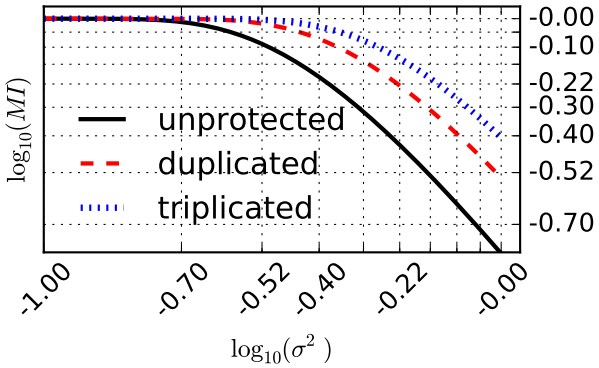


Figure 5.1: MI of instruction n-plication

From Figure 5.1 we derive the following three conclusions. First, we observe that n-plication (for $n > 1$) shifts the MI-curve to the right, i.e. the FI countermeasure produces repeated leakages which have a direct impact on the side-channel security of the implementation. Second, we note that a naively implemented ID may translate to even more than two assembly instructions that manipulate the same value. For instance, a high-level compiler cannot always guarantee the exact amount of instructions, leading to more hazardous repetitions and curve shifts. Third, it follows that a countermeasure designer needs to balance the need for side-channel resistance and FI resistance by fine-tuning the parameter n according to requirements.

5.2.2 Theoretical Evaluation of Infection

It is important to point out that, not only straightforward instruction duplication, but a wide variety of FI countermeasures rely on some form of spatio-temporal redundancy. For instance, detection methods such as full/partial/encrypt-decrypt duplication and comparison of a cipher [135] produce repetitions of intermediate values that are exploitable by the side-channel adversary. Thus, an MI-based evaluation of duplication and comparison is identical to Figure 5.1. Similarly, countermeasures that rely on particular error detection/correction codes [139] also introduce redundancy that has already been evaluated in the side-channel context by Regazzoni et al. [175]. In this section, we expand in the same direction and examine the interaction between side-channel analysis and the more recent infective countermeasure [210][1]. Specifically,

[1]Infective countermeasures in this [210] section do not pertain to the modular arithmetic infective techniques used by Rauzy et al. [173]

we demonstrate how the application of a Hidden Markov Model [77,172] in a low-noise setting can render infective countermeasures equivalent to ID from a side-channel point-of-view. Still, such countermeasures require a 2-step process involving first the HMM attack and (if the first step is successful) the ID-like exploitation.

Infective countermeasures were developed as a solution to the vulnerabilities of the duplicate and compare methods [116]. Instead of vulnerable comparisons, infection diffuses the effect of faults in order to make the ciphertext unexploitable. Several variants of the infective countermeasure exist, so this section focuses on the countermeasure of Tupsamudre et al. [210], which has been proven secure against DFA [163], given that the adversary cannot subvert the control flow and that certain fault models are not applicable [26]. The countermeasure is shown in Algorithm 2. Analytically, infection alternates between real, redundant and dummy cipher rounds (step 8). It requires an r bit random number $rstr$ (step 3), consisting of $2n$ 1's that trigger computation rounds (redundant or real) and $(r - 2n)$ 0's that trigger dummy rounds (steps 5-7). In the event of FI, the difference is detected via function $BLFN : size(R) \rightarrow 1$, where $BLFN(0) = 0$ and $BLFN(x) = 1, \forall x \neq 0$. The error is propagated via step 11.

Algorithm 2: Infection Tupsamudre et al. [210]

Input: Plaintext P, key K, round j key k^j, $\forall j = 1, \ldots, n$, n number of rounds, dummy plaintext β, dummy round key k^0

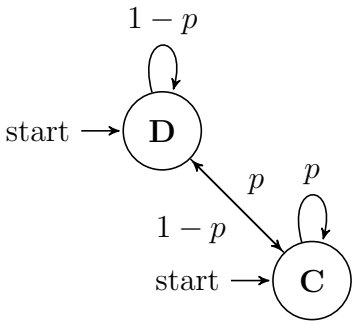
Output: Ciphertext C=Cipher(P, K)

1 Real $R_0 \leftarrow P$, Redundant $R_1 \leftarrow P$, Dummy $R_2 \leftarrow \beta$
2 $i \leftarrow 1$
3 $rstr \in_R \{0,1\}^r$ ▷ //r random bits
4 **for** $q = 1$ *until* r **do**
5 $\quad \lambda \leftarrow rstr[q]$
6 $\quad \kappa \leftarrow (i \wedge \lambda) \oplus 2(\neg\lambda)$
7 $\quad n \leftarrow \lambda\lceil i/2 \rceil$
8 $\quad R_k \leftarrow RoundFunction(R_k, k^n)$
9 $\quad \gamma = \lambda(\neg(i \wedge 1)) \cdot BLFN(R_0 \oplus R_1)$
10 $\quad \delta \leftarrow (\neg\lambda) \cdot BLFN(R_2 \oplus \beta)$
11 $\quad R_0 \leftarrow (\neg(\gamma \vee \delta) \cdot R_0) \oplus ((\gamma \vee \delta) \cdot R_2)$
12 $\quad i \leftarrow i + \lambda$
13 $\quad q \leftarrow q + 1$
14 **end**
15 **return** R_0 ;

From a side-channel perspective, the infective countermeasure can be viewed as a random sequence of r round functions, where only the $2n$ computation rounds are useful for exploitation. Thus, the objective of the side-channel adversary is to uncover the hidden sequence of rounds and to isolate the useful ones. Subsequently, one can exploit

e.g. the first redundant and first real round together via averaging, which is identical to the afore-mentioned exploitation of ID. Distinguishing effectively dummy rounds from computational ones is non-trivial, especially when extra randomization steps are involved [88]. However, the presence of control logic in the infective countermeasure such as variables λ, n and κ can emit noisy side-channel information about the sequence of rounds. We model such leakage as $\mathbf{L}_c = [\Lambda, Z, K] + \mathcal{N}(0, \mathbf{\Sigma})$, where the deterministic part $[\Lambda, Z, K]$ is defined over $\{0,1\}^3$ and $\mathcal{N}(0, \mathbf{\Sigma})$ denotes 3-dimensional noise vector with zero mean and diagonal covariance matrix $\mathbf{\Sigma}$ (homoscedastic noise).

The suggested HMM is constructed the following way. We encode the main loop of Algorithm 2 using two states, i.e. at a given time t, the state $s_t = i \in \{C, D\}$, where C corresponds to a computational round and D to a dummy round. The transitions in the sequence of states is described by matrix T, where $T_{i,j} = Pr(s_{t+1} = j | s_t = i)$. Figure 5.2 shows the state diagram and the probabilities for matrix T, namely $p = 2n/r$. We note that it is possible to unroll the loop and use additional states to describe the transitions, such that we can fine-tune the probabilities. However, we opt for such simple representation to minimize the model's data complexity. In the HMM, the round sequence $\mathbf{s} = [s_1, \ldots, s_r]$ is unknown, but the adversary is assisted by leakage observations $[\mathbf{l}_c^{t=1}, \ldots, \mathbf{l}_c^{t=r}]$. To exploit the observations, the HMM associates every state $i \in \{C, D\}$ with an estimated emission probability function, i.e. emission $e_i(\mathbf{l}_c^t) = Pr(\mathbf{l}_c^t | s_t = i)$.



(a) State diagram.

$$T = \begin{array}{c} \\ \mathbf{C} \\ \mathbf{D} \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{C} & \mathbf{D} \end{array} \\ \begin{pmatrix} p & 1-p \\ p & 1-p \end{pmatrix} \end{array}$$

$$T_{pr} = \begin{array}{c} \begin{array}{cc} \mathbf{C} & \mathbf{D} \end{array} \\ \begin{pmatrix} p & 1-p \end{pmatrix} \end{array}$$

(b) Probability matrices.

Figure 5.2: The infection Markov model describing the states, transition probabilities T and prior probabilities T_{pr}.

Having established the HMM for our scenario, we perform a simulated experiment where we try to identify the round sequence for a gradually increasing noise level. The simulated sequence contains 22 computational rounds and 78 dummy rounds, i.e. it corresponds to a computation of AES-128 using infection with $r = 100$. For every noise level we apply the Viterbi algorithm [218], which can recover the most probable sequence $\mathbf{s}$ of length r, while factoring in the leakage observations $\mathbf{l}_c^{t=1\ldots r}$ and the transition probabilities of T. The simulation (Figure 5.3) shows that for fairly small noise levels (e.g. $\sigma < 0.3$) we are able to uncover the hidden sequence with high probability, making the side-channel exploitation of infection equivalent to the exploitation of instruction duplication.

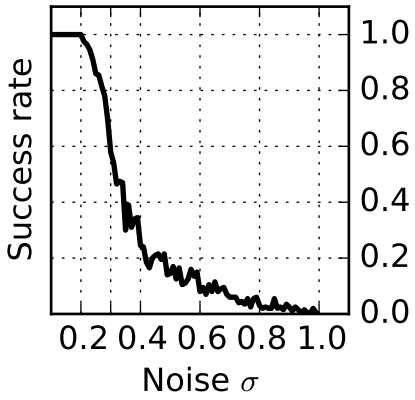


Figure 5.3: Success rate of HMM-based sequence detection vs. noise level σ. Note that for $\sigma < 0.3$ we are able to detect the most probable sequence with high confidence.

5.2.3 Practical Side-Channel Evaluation

In this section, we apply the exploitation techniques of Section 5.2.1 in our experimental setup that protects an AES-128 implementation using ID. We verify the technique's applicability to real-world scenarios by showing their increased efficiency compared to standard SCA methods.

5.2.3.1 Experimental Setup

We use an AVR XMega as the main target for our SCA experiments and we collect power traces using the open-source ChipWhisperer product[2]. The clock frequency of the target is 7.3728 MHz and we sample the power consumption of the target 4 times per clock cycle approximately, at 30 MSamples/sec. The device-under-test can be seen in Figure 5.4.

We use three different code patterns (Table 5.1) to evaluate the interaction between SCA and ID in different scenarios. Listing A and B demonstrate how ID affects different instructions, namely instructions `eor` and `ld` respectively. Listing C showcases the duplicated key addition and Sbox parts of a lookup-table-based AES implementation.

[2] `https://newae.com/tools/chipwhisperer/`

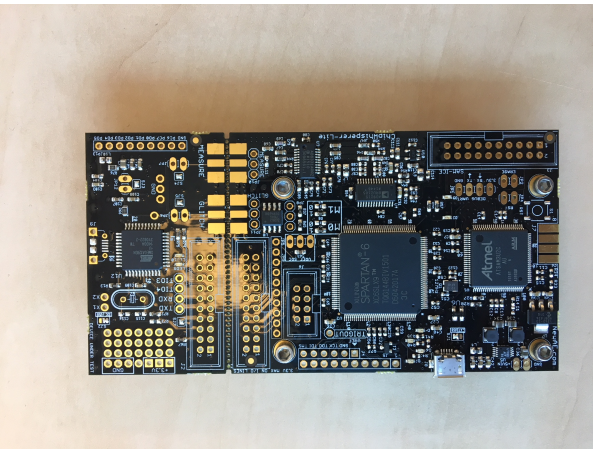


Figure 5.4: ChipWhisperer board, using AVR XMEGA128D4 for device-under-test.

A	`eor r10 , r17`
B	`eor r10 , Y`

C	`eor r9 , r17`
	`add r28, r9`
	`ld r10, Y`

Table 5.1: Code snippets that are duplicated and then used for the SCA experiments. `Y` is the output buffer and `r17` contains the hardcoded secret key.

5.2.3.2 Horizontal Exploitation using CPA

For the afore-mentioned patterns, we perform an experimental evaluation where we put forward a variant of the stardard CPA [41]. In the case of n-plication, we involve a horizontal averaging pre-processing strategy as follows.

1. Locate the intervals pertaining to the n different repeated leakages. In every interval, heuristically select the point in time with the highest correlation to the targeted key-dependent value, obtaining vector $\mathbf{l} = [l^{t=1}, \ldots, l^{t=n}]$.

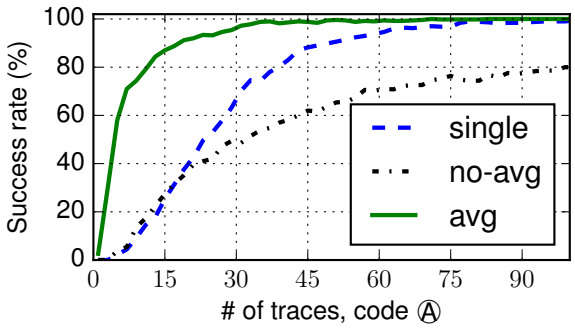
2. For every vector $\mathbf{l}$ compute the average value $\bar{l} = (1/n) * \sum_{t=1}^{n} l^t$, thus reducing the noise level.

3. Perform CPA using the averaged values $(\bar{l})$.

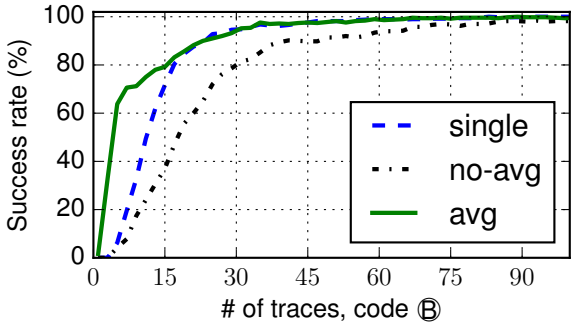
In Figure 5.5, we observe how the averaged CPA using a Hamming weight model outperforms naive CPA that ignores horizontal leakage, since it requires less traces to converge. Thus, the theoretical results of Section 5.2.1 are confirmed in practice and we conclude that horizontal averaging rejects noise. In addition, the difference between the naive CPA on the original code and averaged CPA on the duplicated code is larger on the duplicated `eor` pattern rather than on the duplicated `ld`. This behavior is attributed to the SNR of `ld/st` instructions, which is significantly higher compared

to the SNR of ALU operations (such as `eor`)[3], since the later do not manipulate the memory bus. As a result, there is less need to reject noise on memory instructions. Last, we observe that a naive CPA attack when ID is in place may be slower to converge due to interference between duplicated consecutive instructions.

Last, we note that although this section views n-plication as a fault tolerance mechanism, the same averaging technique can be applied when n-plication is used as a fault detection mechanism. In the latter case the instruction stream is the same as before, therefore, the side channel is amplified in a similar fashion.



(a) Code listing A.



(b) Code listing B.

Figure 5.5: Success rate of the CPA attack for listings A and B. *single* denotes CPA on the original code. On duplicated code, *no-avg* denotes the naive CPA and *avg* denotes CPA with averaging.

5.2.3.3 Horizontal Exploitation using Template Attacks

In order to fully exploit the available horizontal leakage, we also use a template-based approach [50,54], which comprises two phases for attacking an AES-128 implementation: a profiling phase, in which templates are built for 256 key candidates of an AES-128 key byte and an extraction phase, where a number of traces are used to recover the unknown key. In our experiments, for the profiling phase, we use 3.2k traces of the device per key candidate and perform dimensionality reduction, selecting Points of Interest via Principal Component Analysis [9]. The sample interval used for

[3]SNR(A)=2.23 and SNR(B)=18.20

PCA compression is selected using the Pearson correlation heuristic. We deployed the following two template attacks. To ensure that the side-channel effect of ID is exploited during the heuristic step of POI selection, the first attack breaks the trace in multiple intervals, each containing a single assembly instruction and performs POI selection in every interval separately (single-interval TA). The second template attack considers the full trace as a single interval and performs POI selection in the whole region (multi-interval TA).

In Figure 5.6, we focus on code pattern C. We perform the CPA attack (naive and averaged) that exploits the duplication of the `ld` instruction computing the Sbox output. Moreover, we perform the multi-interval and single-interval template attacks. We observe that both template attacks achieve similar performance and surpass the averaged CPA. Thus, we verify the applicability of templates in a horizontal context and conclude that they constitute an optimized way to exploit repeated leakages. We note that template attacks are inherently multivariate and may often require an extensive profiling phase to effectively characterize the model. On the other hand, averaged CPA compresses multiple samples, i.e. it is a univariate technique with a less informative model compared to templates, yet it has the upside of being faster to train and compute.

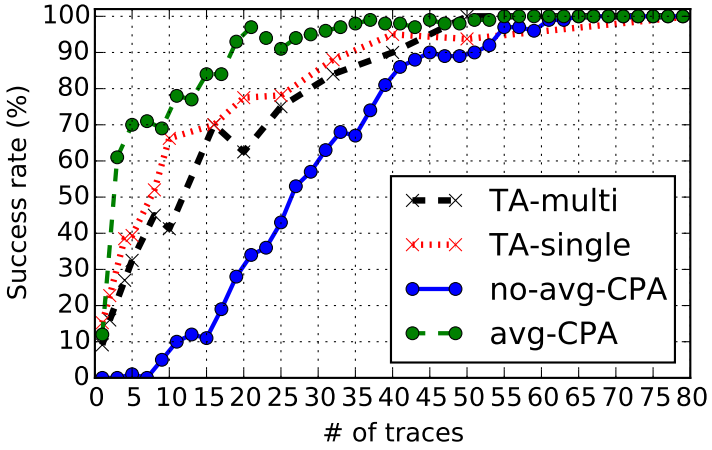


Figure 5.6: CPA vs. Template Attack on code listing C.

5.3 SCA Evaluation of FRIET

Having concluded the side-channel evaluation of FI countermeasures (a posteriori redundancy), this section shifts focus to the side-channel evaluation of FRIET, a FI-resistant cipher (a priori redundancy). Section 5.3.1 provides a description of the cipher and its fault-detecting properties. Continuing, section 5.3.2 performs a horizontal side-channel evaluation of FRIET using a Soft-Analytical Side-Channel Attack. The attack manages to aggregate the leakage from the various FRIET components and can gauge the side-channel impact of its custom fault-detecting structure.

5.3.1 FRIET Cipher Design

In this section, we describe the two structures related to FRIET. The first structure describes the core FRIET permutation and does not include FI-detecting capabilities. We refer to this as the compact FRIET round. The second structure, also referred to as code-abiding FRIET round has an embedded a parity code [4,3,2]. The protection offered by the parity code embedding is that faults in the computation are likely to lead to a decoding error. In this case, the embedded code guarantees the detection of any single fault in the computation.

Compact FRIET iteratively applies a round function on a state of 384 bits divided in three limbs a, b and c of 128 bits. The round function R_i is composed of 6 steps:

- Two non-native limb transpositions τ_1 and τ_2
- Round constant addition δ_i: a limb adaptation
- Two mixing steps μ_1 and μ_2 that are limb adaptations
- A non-linear step ξ, also a limb adaptation

The nominal number of rounds of FRIET is 16. We specify the FRIET permutation in Algorithm 3 using following notation. We illustrate the compact FRIET round function in Figure 5.7.

- $x \oplus y$, the exclusive *or* (XOR) of limbs x and y,
- $x \wedge y$, the bitwise logical *AND* of limbs x and y,
- $k \gg n$, the logical shift to the right by offset n of the 32-bit constant k.
- $x \lll n$, the cyclic shift to the left by offset n of limb x.

We define a linear indexing of the state with i ranging from 0 to 383. a contains the bits with indices 0 to 127, b those with indices from 128 to 255 and c those from 256 to 384.

Continuing, we embed FRIET with the parity code [4, 3, 2], creating code-abiding FRIET that is now enhanced with fault-detection capabilities. For this purpose, we need a checksum limb d and after every step we have $d = a \oplus b \oplus c$. The illustration of the round function of code-abiding FRIET is visible in Figure 5.8.

Finally, we can compare code-abiding FRIET with the simplest code that can detect single-limb faults: the straightforward code $[2, 1, 2]$ where each bit is duplicated. This code doubles both the state size and the computational cost. We provide a comparison in Table 5.2 between compact FRIET, code-abiding FRIET and duplicated FRIET. We observe that although duplicated FRIET and code-abiding FRIET are equivalent from an FI point of view, duplication is a worse option computationally. The same trend applies when comparing the SCA-resistance of code-abiding FRIET with duplicated FRIET, as we will demonstrate in the next section.

Algorithm 3: Compact FRIET

Input: $a, b, c \in \{0,1\}^{128}$

Output: $(a', b', c') = FRIET(a, b, c)$

1 **for** i 0 *until* 15 **do**

2 $\quad (a, b, c) \leftarrow R_i(a, b, c)$

3 **end**

4 Here R_i is specified by the following sequence of steps:

$$
\begin{array}{lcll}
c & \leftarrow & c \oplus (\texttt{0x00}^{12} | \texttt{0xf930b51d} \gg i) & \delta_i \\
(a, b, c) & \leftarrow & (a \oplus b \oplus c, c, a) & \tau_1 \\
b & \leftarrow & b \oplus (c \lll 1) & \mu_1 \\
c & \leftarrow & c \oplus (b \lll 80) & \mu_2 \\
(a, b, c) & \leftarrow & (a, a \oplus b \oplus c, c) & \tau_2 \\
a & \leftarrow & a \oplus ((b \lll 36) \wedge (c \lll 67)) & \xi
\end{array}
$$

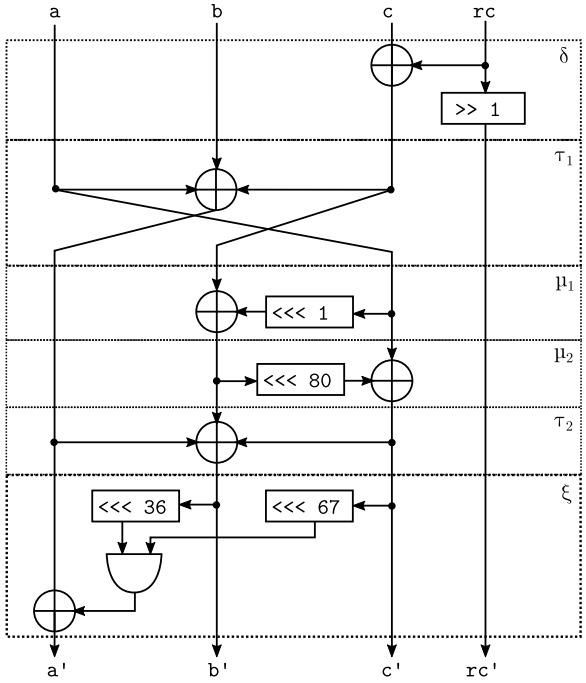


Figure 5.7: Compact FRIET round function R_i

FRIET version	**# limb operations per round**				**total**
	XOR	**rotation**	**AND**	**shift**	**# limbs**
compact	8	4	1	1	3
code-abiding	8	8	2	2	4
duplicated	16	8	2	2	6

Table 5.2: Comparison of computational and storage cost

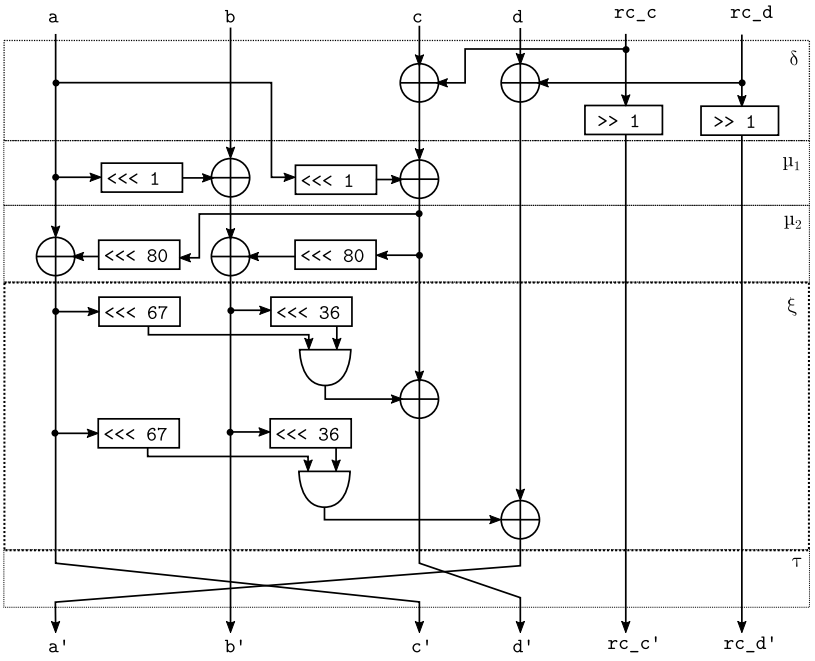


Figure 5.8: Round of code-abiding FRIET

5.3.2 FRIET Side-Channel Evaluation

Chapter 5.2 demonstrated clearly that that standard side channel attacks such such as naive univariate CPA are often incapable of exploiting redundant information added by fault countermeasures. To offer a holistic security analysis of code-abiding FRIET, it is necessary to assess its security with respect to side-channel analysis in a concrete manner. Section 5.2 has already demonstrated that any form of repetitions or redundancy increases the exploitable side-channel information. As a result, in the case of the FRIET fault-detecting permutation, the redundant 128-bit limb d can enhance the available leakage and makes key recovery easier.

Like Section 5.2, we need to encompass all available leakage in our analysis in order to provide a fully-fledged evaluation of a priori redundancy. Unlike, Section 5.2, enhanced side-channel attacks such as correlation power analysis with averaging and template attacks that target a single intermediate value may fail to fully capture the leakage of the embedded code. The fact that such attacks are not able to reveal the full picture, exacerbates the need for more concrete evaluation tools. To tackle this issue, we use Soft Analytical Side-Channel Attacks by Veyrat-Charvillon et al. [215] and Le Bouder et al. [131] in order to attack FRIET effectively. This particular attack is able to exploit the *entire structure* of a cipher/permutation and it can naturally integrate the added redundancy into the side-channel evaluation.

We investigated the impact of the parity limb d and code-abiding round on the SCA vulnerability of FRIET with the Soft Analytical Side Channel Attacks [215]. SASCA is a horizontal type of side channel attack based on the Belief Propagation algorithm [128]. The structure of the SASCA allows exploitation of leakages of any instructions/gates and for our case it can also take advantage of the checksum-limb

(up to XOR limitation studied in [93, 100]).

Our SASCA evaluation has the two following goals.

- Assess the effect of the fault-detecting capabilities to the side channel leakage of FRIET.
- Compare the side channel leakage of code-abiding FRIET with that of a FRIET implementation with simple duplication.

We construct two bipartite graphs following methodology introduced in [215], we concentrate on 1-bit and leakage of the first round (the diffusion of FRIET made propagation across round difficult as highlighted in [93]). The first graph corresponds to compact implementation and the second one to code-abiding implementation. The duplication method is analyzed by averaging the two leakages per intermediate value and use the compact implementation graph as representation. Graphical representation of the graphs are given in Figures 5.9 and 5.10.

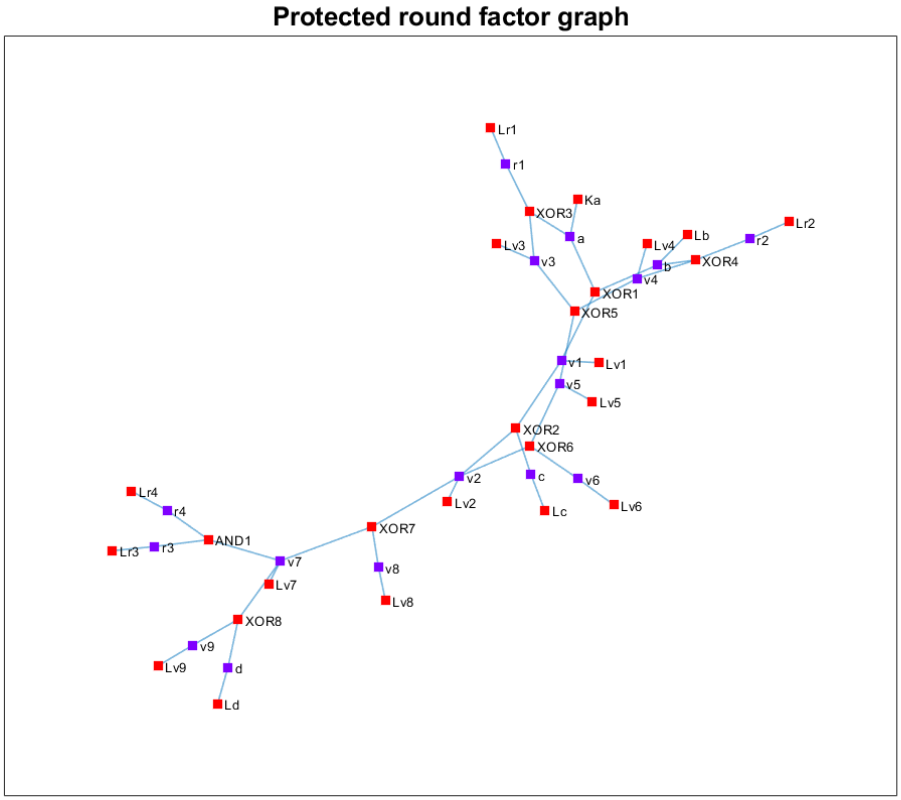


Figure 5.9: Bipartite graph of code-abiding implementation.

We simulate the leakage measurements of each 1-bit intermediate variable v using a Normal distribution $\mathcal{N}(v, \sigma^2)$, where the mean is the identity leakage function of the

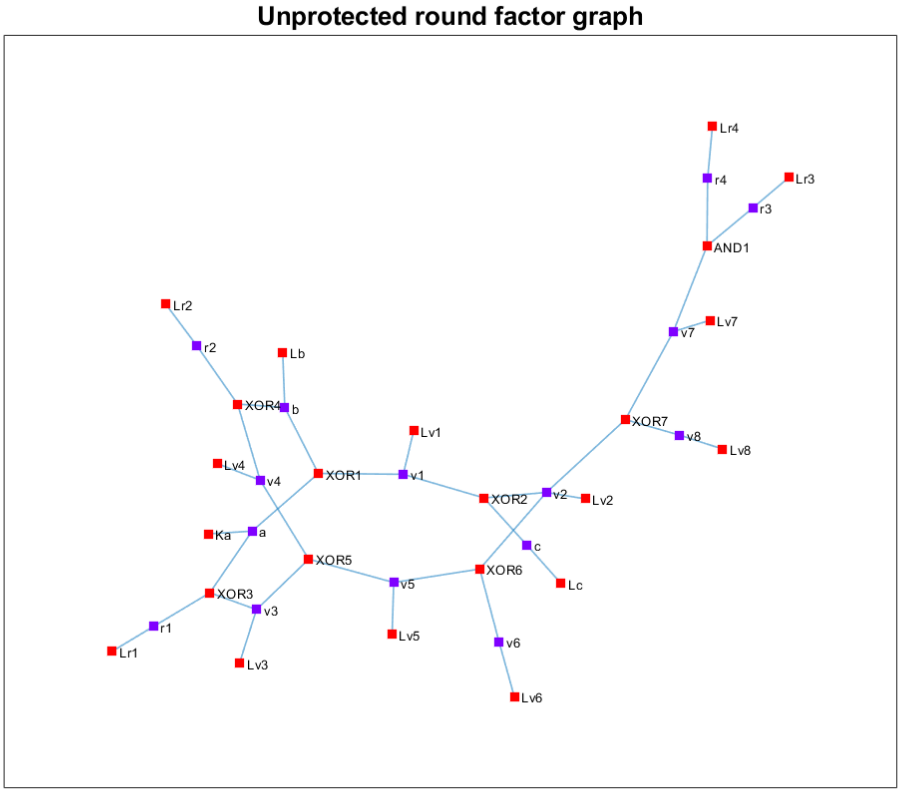


Figure 5.10: Bipartite graph of compact implementation.

variable and the standard deviation σ is the same for all variables. The goal of the attack is to retrieve the value of the bit of the initial value of limb b (remark attacks are similar for other bits, and can be recovery with independent attacks in order to reduce computational cost of SASCA).

Figure 5.11 showcases the average success rate of the different simulated attacks over 500 experiments for $\sigma = 1$ in function of the number of traces used for the attack. Analyzing how fast the different success rates converge to 1, we can derive three core observations.

- First, we conclude from the plot that every version of BP can exploit more information than a straightforward template attack that focuses solely on a single intermediate value and does not factor in the cipher's structure. On the contrary, BP learn the leakage of multiple intermediate values and joins this via message propagation. Thus, we stress the necessity for horizontal exploitation tools such as SASCA, due to the limited effectiveness of single-intermediate statistical templates, which may result in misleading conclusions.

- Second we see that BP on code-abiding FRIET converges faster than BP on the compact FRIET. Thus we are able to observe and quantify the extra leakage penalty that is incurred by the extension.

- Third, we perform a comparison between code-abiding FRIET and duplicated FRIET. We see that the BP attack on code-abiding FRIET structure converges slower than that on duplicated FRIET. Hence extension leads to less exploitable leakage than duplication. As a result, considering SCA and FI jointly, code-abiding FRIET offers a better overall security level than duplicated FRIET.

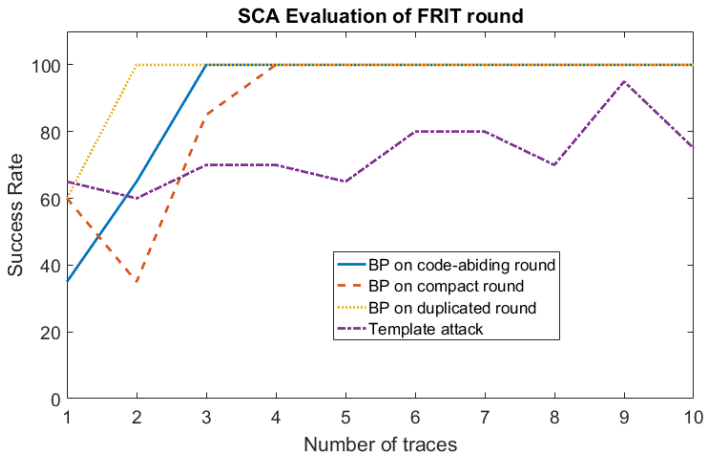


Figure 5.11: Success rate of simulated SASCA

5.4 Conclusions & Future Directions

This chapter has performed a detailed investigation of fault injection resistance and its impact on side-channel security. As a first outcome, we confirm that these two concepts are in fact opposing forces in hardware security. We also confirm that SCA gets enhanced when FI protection is in place. This notion persists throughout various FI countermeasures (n-plication, infection), as well as FI-resistant cipher designs. Finally, we confirm the practical applicability of FRIET, a FI-resistant cipher, since we show that not only is it computationally better compared to duplication methods but it also emits less side-channel information.

The conclusions of this chapter bear close resemblance to the conclusions of Chapter 4. Much like RRM/RRS, this chapter explored tradeoffs between randomness and side-channel security, this chapter establishes the tradeoff between fault injection resistance and side-channel security. Again, this opens up new design options for the countermeasure designer, who can use the toolset of this chapter in order to fine-tune a device according to requirements.

Regarding future work, the first natural step is working towards an integrated MI framework that encompasses SCA, FI and RNG as design parameters. We strongly believe that the plethora of interactions identified in Chapters 4, 5 need to be actively used during the early design stages of a secure device, resulting in optimized constructs.

Concerning FI-resistant cipher designs, we note that FRIET is the first step in this direction. Future work can strive towards developing a cipher design that has stronger build-in FI-resistance. Next generation designs could consider multiple faults and different fault models to protect against. Primary open targets are instruction-skipping faults, as well as ineffective faults [74]. In the same wavelength, we note that we do not often encounter cases were FI resistance is a product requirement, while SCA resistance is out of scope. Thus, future designs can work towards merging both FI and SCA requirements in a single cost-effective cipher structure.

Chapter 6

The Location Leakage Dimension

"I don't like having to play catch-up in security, but we seem doomed to keep doing so."

Bruce Schneier, 2010

So far, Chapters 3, 4 and 5 examine well-known vulnerabilities such as standard side-channel and fault injection attacks. Siding with the defender, they implement and optimize countermeasures, while being mindful of hazardous interactions between them. However, a constant drive in hardware security research is the discovery of unknown vulnerabilities (often called "zero-day" exploits), which reveal a new attack dimension and bypass existing protection mechanisms. Playing catch-up, the industry is forced to react swiftly and patch vulnerable products. Unfortunately, effective patching of hardware implementations is difficult, if not impossible, and can lead to expensive ad-hoc solutions and mass product recalls. Even after the initial turmoil caused by a new exploit, the community usually comes up with focused countermeasures which, once again, interact with existing ones in unexpected ways. This detrimental function creep resembles the interactions between SCA and FI countermeasures of Chapter 5 and can reduce overall security. We do acknowledge that the a priori protection against zero-day vulnerabilities is very hard to achieve. However, we stress that once such an exploit is identified, we must strive to adequately *model* it and *link* it to current attacks and countermeasures, facilitating quick reactions that do not weaken existing protection.

This chapter's goal is to effectively investigate a more recent exploit via statistical modeling and subsequently protect against it, while being mindful of existing attacks and countermeasures. To this end, it revisits a fairly recent vulnerability, namely location-based side-channel leakage. Examined in multiple contexts by Sugawara et al. [204], Heyszl et al. [105] and Messerges et al. [150], this potent yet uncommon form of side-channel leakage originates from the fact that chip structures such as memory and register file emit distinctive information when accessed. This new attack dimension can bypass protected implementations that are well-equipped to withstand the more prevalent data-based SCA. Analytically, this chapter first crafts an adequate theoretical *model* for this recently identified exploit. Second, it performs an in-depth practical evaluation using both standard statistical methods [50] and novel modeling techniques [138]. Third, it examines countermeasures against location-based leakage

in the context of public key cryptography. *Linking* the new location-based SCA with standard, data-based SCA leads to a hybrid attack and prompts a countermeasure analysis with both location-based and data-based attacks in mind.

This chapter is based on work published/submitted in [7, 8, 209] and it deploys microprobing evaluation setups, similarly to earlier work by Heyzl et al. [105]. While exploring location as a new leakage "dimension", we observe that we seem indeed "doomed" to keep patching our designs with new countermeasures that account for unforeseen exploits. In spite of this ceaseless patch cycle, we remain optimistic and work towards swift reactions that can model the zero-day exploit, link it to our existing protection mechanisms and lead to quick yet solid mitigation that does not deteriorate our existing protection. The main points of the chapter are summarized below.

- This chapter examines the often overlooked location-leakage attacks and provides a theoretical model to describe its effects. The model marks the first step towards deeper understanding of such leakage effects and prompts closer examination of a device's physical layer.

- This chapter challenges the current statistical evaluation techniques such as Pearson correlation and template attacks by putting forward deep learning techniques for the location-based side-channel attack. Following the recent research trends, we opt to use neural network classifiers and assess their potential and limitations.

- Like Chapters 4 and 5, this chapter identifies interactions between different types of countermeasures (data and location countermeasures) and analyzes them jointly. The presented hybrid attacks confirms yet again the need for a holistic approach in countermeasure design that factors in the various attack dimensions that are available to the adversary.

6.1 Introduction

Location-based leakage is a fairly new leakage "dimension" that rises in many practical scenarios, despite being a less common target of side-channel analysis. It stems from the fact that chip components such as registers, memory regions or other storage units exhibit leakage when accessed and such leakage is identifiable and data-independent. Thus, the power or EM side-channel potentially conveys information about the *location* of the accessed component, i.e. it can reveal the particular register or memory address that has been accessed, regardless of the data stored in it. If there exists any dependence between the secret key and the location of the activated component, then a side-channel adversary can exploit it to his advantage and recover the key. Although such attacks remain possible using standard power/EM equipment, they were largely assisted by the advent of near-field microprobes, which have the capability to isolate small regions of a chip surface and enable precise measurements with high spatial resolution. Naturally, the appearance of yet another attack surface prompted research, which unfortunately is not often carried in modern microcontrollers and is not always linked to existing protection mechanisms. The discrepancy between the high potential of location-based SCA and the small amount of attention it has received so far motivates the work in this chapter.

6.1.1 Chapter Contribution

In this chapter, we provide a simple spatial model that partially captures the effect of location-based leakages. The model is motivated by experimental data observed in the SRAM of an ARM Cortex-M4 microcontroller. Subsequently, we use the newly established model to simulate different theoretical scenarios that enhance or diminish location-based leakage. We investigate the security of every scenario using the perceived information metric [178].

Continuing, we perform the first practical location-based attack on the SRAM of a modern ARM Cortex-M4, using difference-of-means, multivariate template attacks [50] and neural network classifiers [138]. Using these techniques, we showcase attacks where it is possible to distinguish consecutive SRAM regions of 128 bytes each with 100% success rate and to distinguish between 256 consecutive SRAM bytes with 32% success rate. As result, we conclude that EM location-based leakages are potent enough to compromise the security of public key systems like elliptic curve cryptography and symmetric key systems like AES implementations, should these systems use SRAM lookup-tables.

Finally, we investigate the recently proposed countermeasure of Boolean exponent splitting [209] from the perspective of location-based attacks. We use again the information-theoretic framework of Standaert et al. [197] and the mutual information metric to perform a security evaluation with a data-based attack and a location-based attack. More importantly, we present for the first time a hybrid attack, where data leakage is combined with location leakage and we analyze the countermeasure's effectiveness against it. The rich interactions between data and location leakage corroborates the need for holistic countermeasures that encompass a wide spectrum of

side-channel attacks.

6.1.2 Previous Work

The research on location-based attacks is slightly fragmented, with attacks identified in symmetric and asymmetric cryptography literature, using various devices that range from ASICs and FPGAs to microcontroller units. We also note that location-based leakages can be found in the literature as *address attacks*, originating from side-channel analysis that exploits memory addressing to recover the secret key. Still, we prefer the term location-based leakage, since this multifaceted attack can manifest in multiple chip areas, including but not limited to memory units.

The work of Sugawara et al. [204] was among the first to demonstrate the presence of location-based leakage in an ASIC. In particular, they show that the power consumption of the chip's SRAM conveys information about the memory address that is being accessed. They refer to this effect as "geometric" leakage since it relates to the memory layout. Similarly, Andrikos et al. [8] performed preliminary analyses using the EM-based location leakage exhibited at the SRAM of an ARM Cortex-M4. The work of Heyszl et al. [105] manages to recover the secret scalar by exploiting the spatial dependencies of the double-and-add-always algorithm for elliptic curve cryptography. The experiments were carried out on a decapsulated FPGA, using near-field microprobes that identify the accessed register. Schlösser et al. [184] use the photonic side-channel in order to recover the exact SRAM location that is accessed during the activation of an AES Sbox lookup table. This location information can assist in key recovery, thus even cases of photonic emission analysis can be classified as location-based leakage. Finally, side-channel countermeasures such as RSM [152] rely on rotating lookup tables to mask the data. Location-based leakage can identify which lookup table is currently under use and potentially weaken the masking countermeasure. Public key cryptography has taken already steps towards identifying and mitigating location-based attacks by deploying algorithmic countermeasures. Messerges et al. [149, 150] and Itoh et al. [112] investigated such attacks on modular exponentiation and scalar multiplication, prompting countermeasures from May et al. [146], Itoh et al. [113] and Izumi et al. [114].

For the sake of clarity, in this chapter we distinguish between "location leakage" and "localized leakage". Location leakage arises when knowing the location of a component (register, memory region, etc.) is assisting towards key recovery. On the contrary, localized leakage arises when the adversary is able to focus on the leakage of a specific (usually small) region of the chip. For example, recovering the memory address accessed during an Sbox lookup implies location leakage. Being able to measure the leakage right on top of a processor's register file implies that the adversary is capturing localized leakage. Note that capturing localized leakage can be useful for data-based attacks as well as for location-based attacks. The works of Unterstein et al. [211], Immler et al. [109] and Specht et al. [194–196] acquire localized leakage via an EM microprobe in order to improve the signal-to-noise ratio of their data-dependent leakage. The work of Heyszl et al. [105] uses the same technique in order to improve the signal-to-noise ratio of their location-dependent leakage.

Again, for the sake of clarity we distinguish between "location leakage" and "address leakage" [113]. In our work, address leakage implies the leakage of addressing mechanisms, e.g. the leakage of the control logic of a storage unit. Such leakage can even be observed far from the storage unit itself, e.g. at memory buses or at the CPU. Location leakage implies the leakage caused by such address leakage and the leakage of the unit itself, which is often observed near it. We refer to the latter as "spatial leakage", i.e. location leakage encapsulates both address-related and spatial effects. For example accessing a table in memory requires indexing and memory addressing in the CPU (address leakage). In addition, accessing causes the memory itself to be activated (spatial leakage). The adversary is usually able to observe both types of leakage and it is often hard to distinguish between them.

6.1.3 Chapter Organization

This chapter is organized as follows. Section 6.2 describes the microprobe-based experimental setup on ARM Cortex-M4, shows a simple location analysis using difference-of-means, and motivates experimentally the spatial part of location leakage. The spatial leakage model is provided in Section 6.3, together with several theoretical scenarios, and an evaluation using the perceived information metric. In Section 6.4 we demonstrate practical location attacks on ARM Cortex-M4 using multivariate normal classifiers (template attacks) and Section 6.5 uses neural network classifiers (convolutional neural networks and multi layer perceptrons) to the same end. Finally Section 6.6 analyzes a side-channel countermeasure for elliptic curve cryptography from the viewpoint of data-based and location-based leakages, culminating in a hybrid attack. We conclude in Section 6.7

6.2 Simple Location-Based Analysis

This section describes a high-precision EM-based setup that is able to detect location leakage on the surface of an ARM Cortex-M4 (Sections 6.2.1, 6.2.2). Having established the capabilities of high-precision microprobe setups, we obtain intuition about the location leakage that is caused by switching circuitry and is observable via EM emissions on the die surface (Section 6.2.3). Throughout this chapter, we concentrate on the following adversarial scenario. The device has implemented a key-dependent cipher operation that uses a lookup-table (for symmetric or asymmetric cryptography) and the adversary aims to infer which part of the table is active, i.e. uncover the location information leading to key recovery.

6.2.1 Setup Description

The main goal of our experimental evaluation is to examine whether it is possible to detect the access to different SRAM regions in a modern ARM-based device. Rephrasing, we examine the device's susceptibility to location-based attacks during e.g. memory lookups for AES or register activation for modular exponentiation and

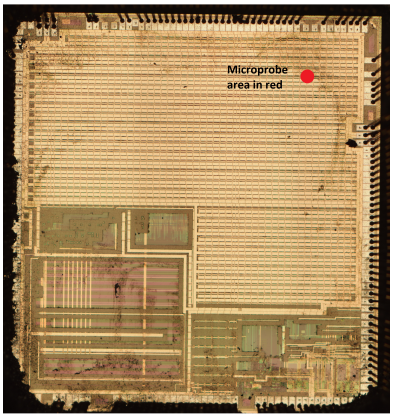


Figure 6.1: The chip surface of the device-under-test (ARM Cortex-M4) after removal of the plastic layer. The approximate area of the ICR HH 100-27 Langer microprobe is shown by the red circle (0.03 mm^2).

scalar multiplication. Our measurement setup consists of a decapsulated Riscure Piñata device[1], using an ARM Cortex-M4 processor on a modified board, fabricated with 90 nm technology. The decapsulated chip surface (roughly 6 $mm^2 \approx 2.4$ $mm \times$ 2.4 mm) is scanned using an ICR HH 100-27 Langer microprobe[2] with diameter of 100 μm (approximately 0.03 mm^2). The scan is performed on a rectangular grid of dimension 300 using the Inspector tooling[3], resulting in 300 $\times$ 300 measurement spots. The near-field probe is moved over the chip surface with the assistance of an XYZ-table with positioning accuracy of 50 μm. At every position of the scan grid, a single measurement is performed, using sampling rate of 1 Gsample/sec and resulting in 170k samples. Due to the complex and non-homogeneous nature of a modern chip, several types of EM emissions are present on the surface, most of which are unrelated to the SRAM location. In this particular case study, the signals of interest were observed in amplitudes of roughly 70 mV, so we set the oscilloscope voltage range accordingly. The signals of interest were identified visually, i.e. in these ranges we could visually distinguish between large SRAM regions. Note that the voltage levels exhibited large fluctuations among different chip regions. Thus, we cannot rule out the possibility that different voltage ranges can also convey location information. In addition, several device peripherals (such as USB communication) have been disabled in order to reduce interference. The decapsulated surface where the scan is performed is visible in Figure 6.1 and the approximate microprobe area is also overlaid on the figure (in red) for comparison. The decapsulated Pinata device and the microprobe setup are visible in Figures 6.2, 6.3.

To effectively cause location-dependent leakage, we perform sequential accesses to a continuous region of 16 KBytes in the SRAM by loading data from all memory positions. The data at all accessed memory positions have been fixed to value zero prior to the experiment in order to remove any data-based leakage. The word size of

[1] `https://tinyurl.com/y9tmnklr`
[2] `https://tinyurl.com/mcd3ntp`
[3] `https://tinyurl.com/jlgfx95`

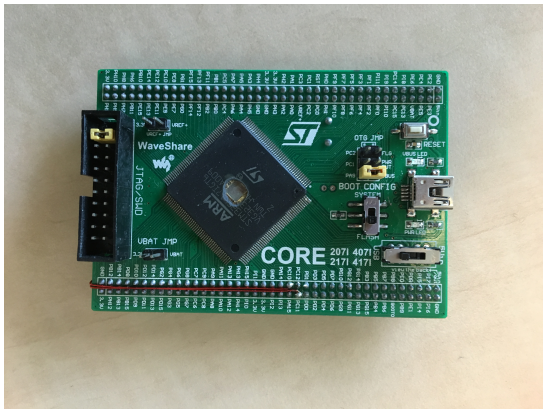


Figure 6.2: Modified Pinata ARM STM32F417IG device.

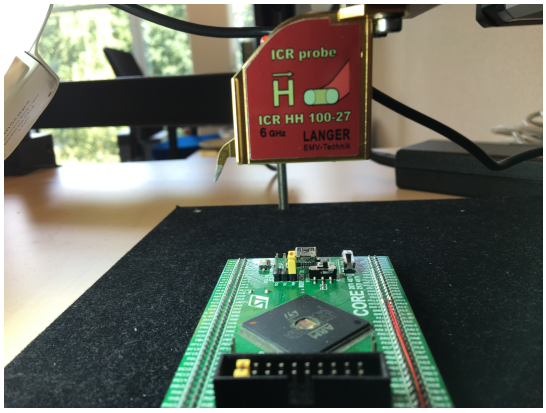


Figure 6.3: Decapsulated Pinata and Langer microprobe ICR HH 100-27 on top.

this ARM architecture is 32 bits, i.e. we accessed 4096 words in memory. We opted to access the SRAM using ARM assembly instead of a high-level language in order to avoid compiler induced optimizations that could alter the side-channel behavior.

6.2.2 Difference-of-Means T-Test

The initial scan measurements were analyzed using a simple difference-of-means test. To demonstrate the presence of location-based leakage, we partitioned every trace (170k samples) into two classes. The first class contains SRAM accesses from the beginning of the memory until word no. 2047 and the second class contains SRAM accesses from word 2048 until word 4096. Each class corresponds to 8 KBytes of SRAM. For every grid position (x, y), we averaged the leakages samples of class 1 and class 2 producing $\bar{l}_{class1} = \frac{1}{85k} \sum_{j=1}^{85k} l_{x,y}^{j}$ and $\bar{l}_{class2} = \frac{1}{85k} \sum_{j=85k}^{170k} l_{x,y}^{j}$ respectively. Continuing, we computed the difference of means $\bar{l}_{class1} - \bar{l}_{class2}$ and we performed a Welch t-test with significance level of 0.1% in order to determine if location-based leakage is present. The results are visible in Figure 6.4, which is focusing on a specific part of the chip surface that exhibits high difference.

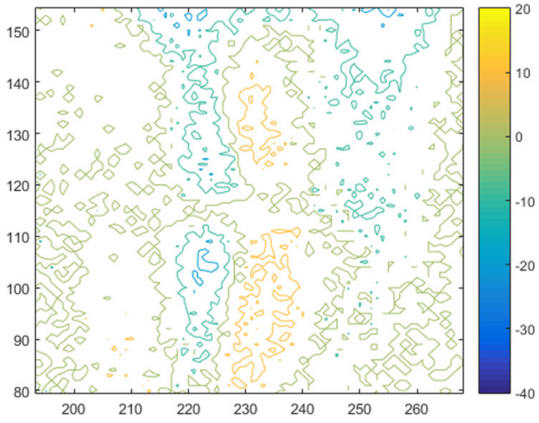


Figure 6.4: Distinguishing two 8 KByte regions of the SRAM with difference-of-means. Yellow region indicates stronger leakage from class 1 while blue region from class 2. Differences below the significance threshold are excluded.

Figure 6.5: Chip surface of ARM Cortex-M4 after removal of the top metal layer. The red rectangular region corresponds to the difference-of-means plot of Figure 6.4, i.e. it shows the location where the highest differences were observed.

6.2.3 Motivating the Spatial Leakage Model

In Figure 6.4 we can observe that location-dependent leakage is indeed present in the ARM Cortex-M4 and it can even be detected through simple visual inspection if memory regions are large enough (8 KBytes). Repeating the same difference-of-means test for SRAM regions of 4 KBytes yields similar results, i.e. the regions

remain visually distinct. In both cases, we observe that these location dependencies demonstrate strong *spatial characteristics*. That is, in Figure 6.4 we see two regions at close proximity (yellow and blue) where the yellow region shows positive difference between class 1 and 2, while the blue region shows negative difference between class 1 and 2. To investigate this proximity, we performed additional chemical etching on the chip surface in order to remove the top metal layer. The result is visible in Figure 6.5.

The different regions (yellow and blue) shown in Figure 6.4 are observed directly above the chip area enclosed by the red rectangle of Figure 6.5. Interestingly, after the removal of the top metal layer, we see that the red rectangular region contains large continuous chip components, possibly indicating that SRAM circuitry is present at this location. This hypothesis is corroborated by the following fact: when we perform difference-of-means test for 4 KByte regions, the yellow and blue regions shrink, indicating that the leakage area is *proportional* to the memory size that is being activated.

The approximate surface area of an SRAM component can be estimated as $a = \frac{m \cdot a_{bit}}{e}$, where m is the number of bits in the memory region, a_{bit} is the area of a single-bit memory cell and e is the array layout efficiency (usually around 70%) [220]. The value of a_{bit} ranges from $600\lambda^2$ to $1000\lambda^2$, where λ is equal to half the feature size, i.e. for the current device-under-test $\lambda = 0.5 * 90\ nm$, thus the area of a 32-bit word is between 55 and 92 μm^2. Likewise, an 8 KByte region of the ARM Cortex-M4 amounts to an area of approximately 0.12 until 0.19 mm^2, depending on the fabrication process. Notably, this area estimation is quite close to the area of the yellow or the blue region of Figure 6.4, or equivalently, approximately half of the red rectangle in Figure 6.5. This observation further supports the relation between SRAM area and location leakage area. Similar spatial characteristics have been observed by Heyszl et al. [105], albeit in the context of FPGA registers.

Our initial conjecture that location leakages exhibit spatial characteristics is backed by experimental evidence that suggest that A) proximity exists between leaky regions and B) the area of leaky regions is approximately proportional to the memory size that we activate. Section 6.3 builds up on these observations and develops a simple spatial model that describes spatial leakage, yet we first need to provide the following disclaimer.

Word of caution. The activation of a memory region can indeed be inferred by observing spatial leakage, which according to experimental data is quite rich in location information. Still, this does not imply that spatial leakage is the sole source of location leakage. It is possible that location information is also revealed through address leakage on the CPU and the memory control logic or buses when they process SRAM addresses, or even by other effects such as imperfect routing [212]. Thus, modeling spatial leakage captures part of the available information and can be considered as the first step towards full modeling of location leakage.

6.3 Location Leakage Model

Unlike the well-established power and EM data leakage models [75,197], high-resolution EM-based location leakage remains less explored. The main reason is the semi-invasive nature of location attacks (often requiring chemical decapsulation), the time-consuming chip surface scanning and the lengthy measurement procedures involved. Still, we maintain that such attacks are increasingly relevant due to the fairly average cost (approx. 15k euros), along with the widespread protection against data leakages [154, 180], which encourages attackers towards different exploitation strategies.

Hence, this section puts forward a theoretical model that describes the spatial part of location leakage on a chip surface, caused by the EM emission of circuit components. The model can be viewed as an extension of the standard data-based model with independent noise to the spatial domain, encapsulating the complexity of surface-scanning experiments. Following the proposed spatial leakage simulation of Section 6.3.1, in conjunction with the information-theoretic framework of Section 6.3.2, we can significantly enhance the design and evaluation cycle of SCA-resistant devices. In particular, our approach allows the countermeasure designer to gauge the amount of experimental work an adversary would need to breach the device using spatial leakage. Thus, the designers can fine-tune any protection mechanisms to provide customized and adequate level of security. At the same time they avoid lengthy design-evaluation cycles as they can capture certain security hazards at an early design stage, using simulation. Thus, the time-consuming leakage certification on the physical device can be carried out at a later stage, once obvious defects have been fixed. Naturally, all simulation-driven models (including this chapter section) have inherent limitations, i.e. they are incapable to describe all the underlying physical phenomena, as we shall see in Section 6.4. Still, avoiding core issues early on, can free up valuable time that evaluators can invest towards device-specific effects such as coupling [58] and leakage combination [161].

6.3.1 Model Definition and Assumptions

Experimental Parameters. We define a side-channel experiment ϵ as any valid instance of the random variable set $\mathcal{E} = \{S, O, G, \mathbf{A}, \mathbf{P}\}$. The experimental parameters are shown in Table 6.1. We designate the experiment's goal to be the acquisition

Parameter	Description	Unit
S	chip surface area	u^2
O	probe area	u^2
G	scan grid dimension	¡no unit¿
A	component areas	vector with 1D entries of u^2
P	component positions	vector with 2D entries of u

Table 6.1: Parameters of simulated experiment

spatial leakage $\mathbf{L}$, i.e. obtain $(\mathbf{L}|\mathcal{E} = \epsilon)$ or $(\mathbf{L}|\epsilon)$ for short. Much like Section 6.2,

the experiment consists of a probe scan over the chip surface in order to distinguish between different components (or regions) and ultimately between different memory addresses, registers, etc. The parameter S denotes the *area of the chip surface* on which we perform measurements, e.g. s can be the whole chip die (6 mm^2) or any smaller surface. Parameter O denotes the *area of the measuring probe* that we use in our experiments, e.g. the area o of the ICR HH 100-27 microprobe is roughly 0.03 mm^2. Typically, we require o to be smaller than s in order to be able to isolate and distinguish different regions on the surface. Continuing, parameter G denotes the *measurement grid dimensions*, i.e. it specifies the resolution of a uniform rectangular array of antennas [213]. In Section 6.2.1 we opted for $g = 300$. Continuing, the vector parameters $\mathbf{A}, \mathbf{P}$ describe the n_c surface components that emit EM-based spatial leakage. The parameter $\mathbf{A} = [A_1, A_2, \ldots, A_{n_c}]$ describes the surface *area occupied by each component*, e.g. in Section 6.2.3 we estimated the area of an 32-bit word component to be at most 92 μm^2. The parameter $\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{n_c}]$ describes the *position of every component* on the chip surface, i.e. $\mathbf{P}_i$ is a two-dimensional vector. For simplicity, we assume the geometry of the surface, probe and components to be square, yet we note that the model can be extended to different geometrical shapes in a straightforward manner. Moreover, we assume that the measuring probe can capture only emissions that are directly beneath it, i.e. it functions like an identity spatial filter with area o.

Control Parameter. Every device can use program code to activate different components of the chip surface, e.g. by accessing different SRAM words through load/store instructions. To describe this, we use an additional control parameter $\mathbf{C}$ that denotes which components (indexed $1, \ldots, n_c$) are accessed during a particular experiment ϵ. Analytically, $\mathbf{C} = [C_1, C_2, \ldots, C_{n_c}]$, where $C_i = 1$ if component i is active during the experiment and $C_i = 0$ if it is inactive; for instance the vector $\mathbf{c} = [0, 1, 0]$ implies that the surface has 3 components ($n_c = 3$) and only component no. 2 is currently active. Note also that in our model only one out of n_c components can be active at a given point in time, since we assume that the ordinary microcontrollers do not support concurrent memory access.[4]. Thus the parameter $\mathbf{c}$ uses the one-hot encoding and we define $\mathbf{v}^i$ as an n_c-dimensional vector where all entries are zero except for the i^{th} entry. For instance, if $n_c = 3$, then $\mathbf{v}^3$ is equal to $[0, 0, 1]$ and it describes the program state where only component no. 3 is active.

In general, we use the notation $(\mathbf{L}|\mathcal{E} = \epsilon, \mathbf{C} = \mathbf{v}^i)$ or equivalently $(\mathbf{L}|\epsilon, \mathbf{v}^i)$ to describe a side-channel experiment ϵ that captures the leakage when the i^{th} component is active. For a location-based attack to be successful, we need to distinguish between two (or more) different components using this spatial leakage. Formally, we need to be able to distinguish between $(\mathbf{L}|\epsilon, \mathbf{v}^i)$ and $(\mathbf{L}|\epsilon, \mathbf{v}^j)$, where $i \neq j$.

Representative Example. To elucidate the model, Figure 6.6 presents an experiment ϵ using parameters $\{s, o, g, \mathbf{a}, \mathbf{p}\}$ equal to $\{25, 3, 2, [0.8, 3], [[0.6, 1.5], [1.6, 4.1]]\}$, where all position parameters are in arbitrary units u and all area parameters are in

[4]Our approach can be easily modified in the case of parallel word processing.

square units u^2. The experiment targets two components ($n_c = 2$) and their position is $[0.6\ u, 1.5\ u]$ and $[1.6\ u, 4.1\ u]$ respectively. The surface area s, probe area o, and component areas a_1 and a_2 are respectively 25 u^2, 3 u^2, 0.8 u^2 and 3 u^2. The dimension g of the measurement grid is 2, resulting in a 2×2 scan and we capture a single measurement (trace) in every grid spot. We use the program code (control parameter) to activate components 1 and 2, generating $(\mathbf{L}|\epsilon, [1, 0])$ and $(\mathbf{L}|\epsilon, [0, 1])$ respectively. Note that in general $(\mathbf{L}|\epsilon, \mathbf{v}^i)$ results in leakage with g^2 dimensions, e.g. $(\mathbf{L}|\epsilon, [1, 0])$ is a 4-dimensional vector. We refer to the leakage measured at any specified position $[x, y]$ as $(L_{[x,y]}|\epsilon, \mathbf{v}^i)$ or simply $L_{[x,y]}$ when the experimental and control parameters are clear from the context.

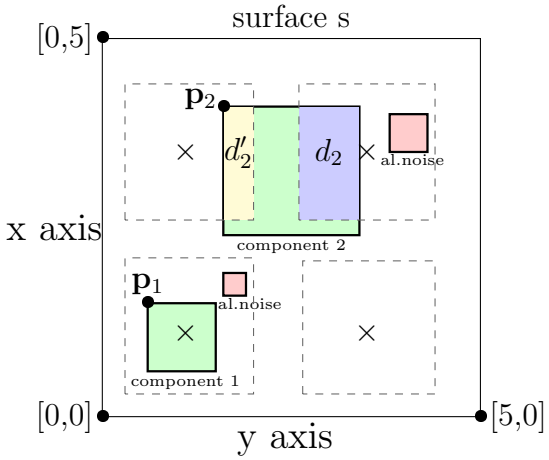


Figure 6.6: Sample experiment ϵ. The $\times$ spots show the measurement points of the 2×2 scan grid. Dashed black-line rectangles enclosing these spots denote the measuring probe area o. Vectors $\mathbf{p}_1, \mathbf{p}_2$ show the position of two components ($n_c = 2$), whose areas (a_1, a_2) are enclosed by the solid black-line rectangles. The blue area d_2 shows the area of component 2 captured by the top-right measurement point and the yellow area d_2' shows the area of component 2 captured by the top-left measurement point.

Independent Noise. In accordance with standard data-based leakage models, we assume that for given parameters $\epsilon, \mathbf{v}^i$, the leakage $L_{[x,y]}$ at any grid position $[x, y]$ consists of a deterministic part $l^{det}_{[x,y]}$, an algorithmic noise part N^{algo} and an electrical noise part N^{el}, thus:

$$L_{[x,y]} = l^{det}_{[x,y]} + N^{algo} + N^{el} \tag{6.1}$$

Deterministic Leakage. We assume that the deterministic part of the leakage $l^{det}_{[x,y]}$ at position $[x, y]$ is caused by the activation (switching behavior) of any component that is captured by the probe at this grid position. Based on the experimental observations of Section 6.2.3, we assume the deterministic leakage to be proportional to the area of the active component located underneath the probe surface, thus:

$$l^{det}_{[x,y]}|\mathbf{v}^i = \begin{cases} 0, \text{ if comp. } i \text{ is not captured at [x,y]} \\ d_i,\ 0 < d_i < a_i, \text{ if comp. } i \text{ is partially captured at [x,y]} \\ a_i, \text{ if comp. } i \text{ is fully captured at [x,y]} \end{cases} \tag{6.2}$$

For example, Figure 6.6 shows that component 1 is fully captured by the probe on the bottom-left grid spot, thus $(l^{det}_{[down,left]}|\mathbf{v}^1) = a_1$. Since no other measurement position can capture component 1, it holds that $(l^{det}|\mathbf{v}^1) = 0$ for the other three grid positions. On the contrary, component 2 is partially captured in two grid positions. Thus, it holds that $(l^{det}_{[up,right]}|\mathbf{v}^2) = d_2$ (blue area), $(l^{det}_{[up,left]}|\mathbf{v}^2) = d_2'$ (yellow area) and zero elsewhere.

Electrical and Algorithmic Noise. We employ the common assumption that the electrical noise N^{el} follows a normal distribution with zero mean and variance σ^2_{el}, i.e. $N^{el} \sim Norm(0, \sigma^2_{el})$. The variance σ^2_{el} is related to the specific device-under-test and measurement apparatus that we use (probe, oscilloscope, amplifier etc.).

The algorithmic noise in our model is caused by components that, like the targeted components, leak underneath the probe on measurement spot of the scan grid. However, unlike our targeted components, they exhibit uniformly random switching activity (equiprobable 'on' and 'off' states) that is independent of the control parameter $\mathbf{c}$. If n_a such components, with area parameter $\mathbf{b} = [b_1, b_2, \ldots, b_{n_a}]$ are located under the probe, then we assume again their leakage to be proportional to the respective captured area. The leakage of these independent, noise-generating components is denoted by N^{algo}_i, $i = 1, \ldots, n_a$. Thus, N^{algo} constitutes of the following sum.

$$N^{algo} = \sum_{i=1}^{n_a} N^{algo}_i, \text{where } N^{algo}_i \sim Unif(\{0, b_i\}) \tag{6.3}$$

The algorithmic noise is highly dependent on the device-under-test, i.e. we could potentially encounter cases where there is little or no random switching activity around the critical (targeted) components, or we may face tightly packed implementations that induce such noise in large quantities. For example, in Figure 6.6 the top-left and bottom-right spots have no algorithmic noise, while the top-right and bottom-left spots contain randomly switching components (red rectangles) that induce noise. Note that the larger the probe area o, the more likely we are to capture leakage from such components.

Algorithmic Noise in Tightly-Packed Surfaces. Since countermeasure designers opt often for algorithmic noise countermeasures, we investigate the statistical variance of N^{algo} for a tightly packed circuit that contains a large number of randomly switching components which try to hide the targeted component. We assume every noise-generating component to have area $b_i \approx d$, where d is the area of the targeted component Since we assume large n_a, both the noise-generating components as well as the targeted component are small w.r.t. the probe size, i.e. $d \ll o$. In a tightly

packed circuit, the probe area o contains roughly $\frac{o}{d}$ randomly switching components, i.e. $n_a \approx \frac{o}{d}$. In this particular scenario, the following formula approximates N^{algo}.

$$N^{algo} = \sum_{i=1}^{n_a} N_i^{algo} = d \cdot \sum_{i=1}^{n_a} B_i = d \cdot A, B_i \sim Bern(0.5) \ , \ A \sim Binomial(n_a, 0.5) \tag{6.4}$$

$$\text{Thus, } N^{algo} \xrightarrow[\text{Theorem}]{\text{Central Limit}} Norm(\frac{d \cdot n_a}{2}, \frac{d \cdot n_a}{4}) \tag{6.5}$$

Using the approximation of the Central Limit Theorem, we see that $Var[N^{algo}] = \frac{d \cdot n_a}{4} = \frac{o}{4}$. Thus, for the tightly-packed, small-component scenario we have established a direct link between the probe area o and the level of algorithmic noise, demonstrating how increasing the probe area induces extra noise.

6.3.2 Information-Theoretic Analysis

The proposed spatial leakage model of Section 6.3.1 is able to simulate the EM emission over a chip surface and provide us with side-channel observables. Due to the complexity of surface-scanning experiments, the model needs to take into account multiple parameters in ϵ (component area, grid size, noise level, etc.), all of which can directly impact our ability to distinguish between different regions.

In order to demonstrate and gauge the impact of the experimental parameters on the side-channel security level, this section introduces an information-theoretic apporach to analyze the following simple location-leakage scenario. Using the model of Section 6.3.1, we simulate the spatial leakage emitted by the ARM Cortex-M4 SRAM, while computing the AES Sbox using a lookup-table (LUT). We choose to simulate this symmetric crypto algorithm (instead of a public key algorithm) due to the higher level of flexibility and granularity that we can display in our experiments. Still, the results are analogous in the case of modular exponentiation or scalar multiplication in asymmetric systems. The ARM Cortex-M4 uses a 32-bit architecture, thus we represent the 256-byte lookup table with 64 words (4 bytes each) stored consecutively in SRAM. The LUT memory region is placed randomly on a chip surface with $s = 0.6$ mm^2. Subsequently, our model can generate the leakage stemming from 64 chip components ($n_c = 64$), where each one occupies surface area pertaining to 4 SRAM bytes. Using the simulated traceset, we perform template attacks [50] after PCA-based dimensionality reduction [9], in order to distinguish between different LUT regions. Note that LUT regions may consist of one or more words, yielding different granularity options to the adversary. Naturally, being able to infer which SRAM region was accessed can substantially reduce the number of AES key candidates. For instance, the adversary may template separately the leakage of all 64 words (high granularity) in order to recover the exact activated word and reduce the possible AES key candidates from 256 to 4. Alternatively, he can opt to partition the LUT e.g. to two regions (words 1 until 32 and words 33 until 64), profile both regions (low granularity), in order to recover the activated 128-byte region and reduce the AES key candidates from 256 to 128.

Formally, at a certain point in time, the microcontroller is able to access only one out of 64 components (high granularity), thus the control variable $\mathbf{c} \in \mathcal{V} = \{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^{64}\}$ and the adversary can observe the leakage of word-sized regions $(\mathbf{L}|\mathbf{C} = \mathbf{v}^i)$, for $i = 1, 2, \ldots, 64$. Alternatively (low granularity), he can focus on $|\mathcal{R}|$ memory regions and partition the set $\mathcal{V}$ to sets $\mathcal{V}^1, \mathcal{V}^2, \ldots, \mathcal{V}^{|\mathcal{R}|}$, where usually $\mathcal{V}^r \subset \mathcal{V}$ and $\mathcal{V}^i \cap \mathcal{V}^j = \emptyset$, for $i \neq j$. We define random variable $R \in \mathcal{R} = \{1, 2, \ldots, k\}$ to denote the activated region and we represent the leakage of region r as $(\mathbf{L}|R = r) = (\mathbf{L}|\mathbf{c} \in \mathcal{V}^r)$. For example, in the high granularity scenario, the adversary observes and profiles $(\mathbf{L}|\mathbf{v}^1), (\mathbf{L}|\mathbf{v}^2), \ldots, (\mathbf{L}|\mathbf{v}^{64})$, while in the low granularity scenario he profiles two regions $(\mathcal{R} = \{1, 2\})$ with $\mathcal{V}^1 = \{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^{32}\}$ and $\mathcal{V}^2 = \{\mathbf{v}^{33}, \mathbf{v}^{34}, \ldots, \mathbf{v}^{64}\}$. Thus he can obtain $(\mathbf{L}|R = 1) = (\mathbf{L}|\mathbf{c} \in \mathcal{V}^1)$ and $(\mathbf{L}|R = 2) = (\mathbf{L}|\mathbf{c} \in \mathcal{V}^2)$.

Having completed the profiling of regions for a certain experiment ϵ, we proceed to the quantification of the location information, using the perceived information metric (PI) [178] that is shown below.

$$PI(\mathbf{L}; R) = H[R] - H_{true,model}[\mathbf{L}|R] = H[R] + \sum_{r \in \mathcal{R}} Pr[r] \int_{\mathbf{l} \in \mathcal{L}^{g^2}} Pr_{true}[\mathbf{l}|r] \cdot log_2 Pr_{model}[r|\mathbf{l}] \, d\mathbf{l}$$

$$\text{where } Pr_{model}[r|\mathbf{l}] = \frac{Pr_{model}[\mathbf{l}|r]}{\sum_{r^* \in \mathcal{R}} Pr_{model}[\mathbf{l}|r^*]}, \; Pr_{true}[\mathbf{l}|r] = \frac{1}{n_{test}}, n_{test} \text{ test set size} \tag{6.6}$$

PI can quantify the amount of information that leakage $\mathbf{L}$ conveys about the activated region R, taking into account the divergence between the real and estimated distributions. Computing PI requires the distribution $Pr_{model}[\mathbf{l}|r]$, i.e. the leakage template that is estimated from the training dataset. In addition, it requires the true leakage distribution $Pr_{true}[\mathbf{l}|r]$, which is unknown and can only be sampled directly from the test dataset. We opt for this metric since it indicates when degraded (under-trained) leakage models are present, due to our choice of experimental parameters. Negative PI values indicate that the trained model is incapable of distinguishing regions, while a positive value indicates a sound model that can lead to classification.

By following the proposed spatial leakage simulation, together with the information-theoretic framework we simulate several leakage scenarios for the data-independent LUT case that emulates the AES Sbox operation. Sections 6.3.2.1 until 6.3.2.5 showcase how different experimental parameters hinder or enhance leakage, offering several design options. To apply the theoretical model in an evaluation context we can simply set our current device SNR to the information-theoretic graphs of the following subsections.

6.3.2.1 Area and number of regions

The first simulation scenario examines the core attack question: using a certain experimental setup with parameters $\epsilon = \{s, o, g, \mathbf{a}, \mathbf{p}\}$, what is the smallest region size that I can distinguish reliably? Rephrasing, we assess how much location information can be extracted from the observed leakage by plotting the $PI(\mathbf{L}; R)$ metric against the

electrical noise variance σ^2_{el} for certain ϵ and $\mathbf{c}$ parameters. We simulate an adversary that distinguishes regions of an AES Sbox lookup-table using the following three LUT partitions of increasing granularity. First, he partitions the 256-byte LUT to 2 regions of 128 bytes each (depicted by the solid line in Figure 6.7). Second, he partitions the LUT to 8 regions of 32 bytes (dashed line) and third to 16 regions of 16 bytes (dotted line). For every partition the adversary profiles the regions' leakage $(\mathbf{L}|R = r) = (\mathbf{L}|\mathbf{c} \in \mathcal{V}^r)$ for $r = 1, 2, \ldots, |\mathcal{R}|$, where $|\mathcal{R}| = 2$ or 8 or 16 and subsequently tries to distinguish. Note that surface $s = 6\ mm^2$, probe size $o = 0.03\ mm^2$ (ICR HH 100-27), feature size 90 nm and $g = 100$, i.e. the scan resolution is 100×100. The component area $a = 92\ \mu m^2$ for all SRAM words and the words are placed adjacent to each other, starting from a random surface position; we denote this as $\mathbf{p} = random$. Along with parameters ϵ and $\mathbf{c}$, we need to include the measurement complexity in our simulation. Thus, we specify the amount of traces measured at every grid spot, resulting in an acquisition of $g^2 \cdot \#traces$. As expected, we observe

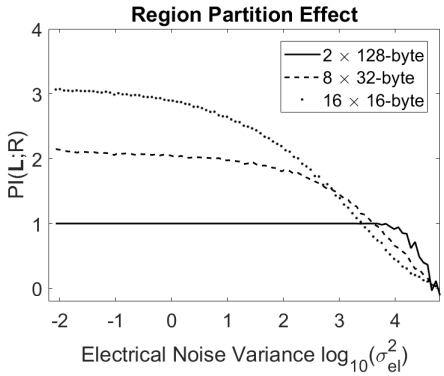


Figure 6.7: Effect of region partitioning of the 256-byte LUT to 2, 8 and 16 regions. Experimental parameters $\epsilon = \{6\ mm^2, 0.03\ mm^2, 100, 92\ \mu m^2, \text{random}\}$, capturing 10 traces per spot for a total of 100k traces.

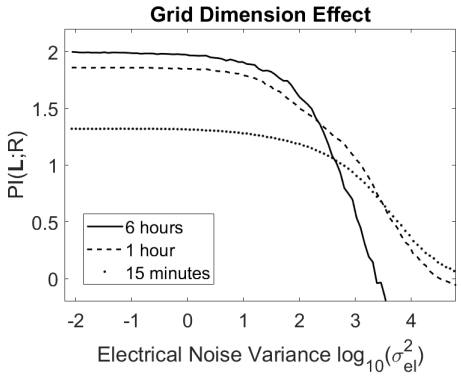


Figure 6.8: Effect of grid dimension $g = 100, 40$ and 20. Parameters $\epsilon = \{6\ mm^2, 0.03\ mm^2, g, 92\ \mu m^2, \text{random}\}$, distinguishing 4 regions of 64 bytes each and using 10, 62 and 250 traces per spot for a total of 100k traces.

that experiments with higher region granularity yield more location information, as shown by the vertical gaps of the PI metric in Figure 6.7. Still, we also observe that smaller regions are harder to distinguish, even for low noise levels. Partitioning to 8 or 16 regions could optimally yield 3 or 4 bits of information respectively, yet the dashed and dotted curves remain well below this limit. Thus, we observe that the adversary may need to improve his experiment ϵ by measuring more traces, using smaller probes or increasing the grid dimension in order to extract the maximum location information.

6.3.2.2 Measurement grid dimension

Any side-channel experiment involving surface scanning can be particularly time-consuming. For instance moving the Langer microprobe between adjacent positions takes approximately 2 seconds, thus the 300×300 surface scan carried out in Section 6.2 takes almost 2 days to conduct. Using the spatial leakage simulation, we are able to specify the grid dimension g and find the minimum scan resolution required to distinguish between certain SRAM regions. In Figure 6.8 we demonstrate the location information captured when conducting scans with resolutions 100×100, 40×40 and 20×20, taking approximately 6 hours, 1 hour and 15 minutes respectively. Across the three simulations we maintain constant data complexity of 100k traces, distributed to grid spots accordingly (10, 62 and 250 traces per spot). Figure 6.8 shows information loss (vertical gap) as the grid dimension is decreasing, i.e. when trying to distinguish 4 regions only the 6 hour-experiment with 100×100 grid is able reach maximum information (2 bits). Notably, we also observe that for larger noise levels, small grid sizes with many traces per spot (dense measurements) are able to outperform larger grid sizes with less traces per spot (spread measurements).

6.3.2.3 Feature size

A common issue encountered in the side-channel literature is the scaling of attacks and countermeasures as devices become more complicated and feature size decreases [118, 144, 153]. This section uses our simple spatial leakage model to describe the effect of feature size on SCA. In particular, it simulates the spatial leakage of SRAM cells fabricated with 180 nm, 120 nm and 90 nm technologies, resulting in bit cell areas of approximately 8 μm^2, 3.5 μm^2 and 2 μm^2. The results are visible in Figure 6.9. Naturally, smaller technology sizes can potentially limit the amount of available information, as they decrease the region's area and force the adversary towards more expensive tooling.

6.3.2.4 Algorithmic noise

This section simulates the countermeasure of spatial algorithmic noise, when implemented on the ARM device. Analytically, we examine the case where the designer is able to place word-sized noise-generating components on the chip surface in order to "blur" the spatial leakage of a targeted region and hinder recovery. The simulation (Figure 6.10) uses formula (6.5) of Section 6.3.1 to approximate the algorithmic noise when the probe captures the leakage of 11 SRAM words, one of which is the target word (and reveals the critical region information) and the ten remaining words are randomly activated at the same time. Note that such an algorithmic noise countermeasure would require parallel accesses to the memory, thus it would be more natural to deploy it on FPGA or ASIC devices. Observing Figure 6.10, we see the algorithmic-noise PI curve (dashed line) shifting to the left of the PI curve without algorithmic noise (solid line). Thus, much like data-based algorithmic noise [198], we see that randomly activating words functions indeed as an SCA countermeasure.

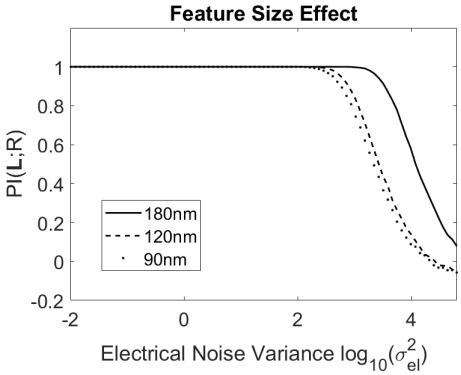


Figure 6.9: Feature size of 180 nm, 120 nm, 90 nm and word area $a = 368\ \mu m^2, 163\ \mu m^2, 92\ \mu m^2$. Parameters $\epsilon = \{6\ mm^2, 0.03\ mm^2, 40, a, \text{random}\}$, for 2 regions of 128 bytes each, 250 measurements per spot for a total of 400k traces.

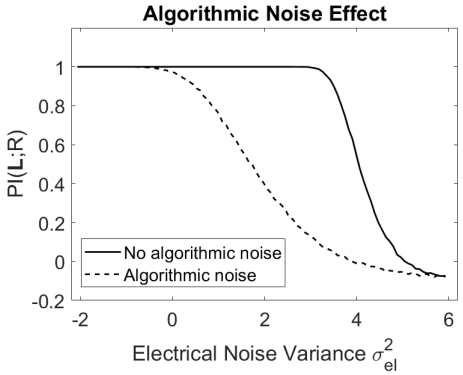


Figure 6.10: Algorithmic noise, using 10 noise-generating words. Parameters $\epsilon = \{6\ mm^2, 0.03\ mm^2, 40, 92\ \mu m^2, \text{random}\}$, for 2 regions of 128 bytes each, 250 measurements per spot for a total of 400k traces.

6.3.2.5 Region proximity and interleaving

Last, we simulate the countermeasure of region proximity and region interleaving on the ARM device, which was initially proposed on a very low design level by He et. al [102] and also considered on a higher level by Heyszl [104]. Analytically, we assume that the designer controls the place-and-route process and can place two memory regions on the chip surface using the following three configurations.

1. Distant placement: the distance between the two regions is set at roughly 1 mm.
2. Close placement: the two regions are adjacent to each other.
3. Interleaved placement: the words of the two regions are interleaved together in a checkered fashion, i.e. the 1st word of the SRAM belongs to the 1st region, the 2nd word to the 2nd region, the 3rd word to the 1st region, etc.

Figure 6.11 demonstrates the effect of different placement choices, confirming the basic intuition that higher proximity is essentially a countermeasure against location-based attacks. The vertical gap in PI between distant, close and interleaved placement shows that as components get closer, the attainable information decreases, forcing the adversary to increase the grid size or use a smaller probe.

6.4 Exploitation Using Statistical Templates

Having established a theoretical model for spatial leakages, we move towards side-channel exploitation with a practical case in mind. In particular, Sections 6.4 and 6.5 exploit the available location-based leakages in the ARM Cortex-M4 so as to infer

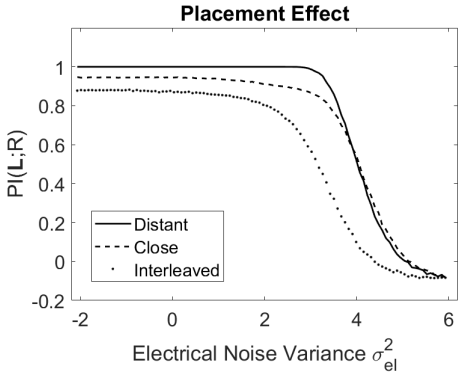


Figure 6.11: Effect of distant, close and interleaved placements (solid, dashed and dotted line). Parameters $\epsilon = \{6\ mm^2, 0.03\ mm^2, 20, 92\ \mu m^2, \text{random}\}$, distinguishing 2 regions of 128 bytes each and using 250 measurements per spot for a total of 100k traces.

the accessed memory position of a 256-byte, data-independent LUT. Note that in the real chip we cannot isolate spatial from address leakage, i.e. we observe location leakage in its entirety. We use the template attack, i.e. we model the leakage using a multivariate normal distribution and attack trying to identify the key, or in our case region r of the SRAM.

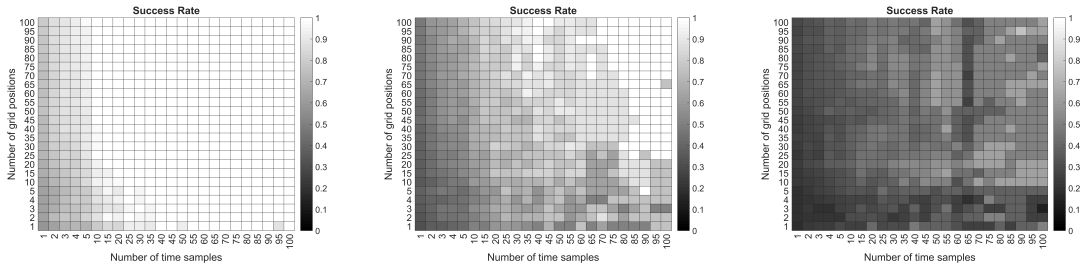
In the our case study, the template attacks focus on identifying which region of the SRAM is being accessed. The leakage vector $(\mathbf{L}|R = r)$ exhibits particularly large dimensionality and can generate a sizeable dataset, even for modest values of the grid dimension g. Thus, we employ dimensionality reduction techniques based on the correlation heuristic so as to detect spatial points of interest (POIs) in the 300 $\times$ 300 grid and use a train-test ratio of 70-30. In addition, when performing template matching, we combine several time samples from the test set together (multi-sample attack), in order to reduce the noise and improve our detection capabilities[5]. In a sizeable dataset the template attacks are particularly demanding w.r.t. computational resources and to tackle this problem, we opt for the improved template formulas proposed by Choudary et al. [54] that use a pooled covariance matrix and numerical speedups. The goal of our template-based evaluation is not only to answer whether location exploitation is possible but also to gauge the effect of the experimental parameters ϵ on the exploitation process. Thus, similarly to Sections 6.3.2.1, 6.3.2.2, 6.3.2.5, we will investigate the effect of region partition, grid dimension and region placement in the real-world scenario. Unfortunately Sections 6.3.2.3 and 6.3.2.4 would require control over the manufacturing process (i.e. several chips of different feature size) or control over regions with algorithmic noise (i.e. parallel memory activation), thus they cannot be tested in our current context. Throughout this section we will engage in comparisons between the theoretical model of Section 6.2 and our real-world attack, i.e. we will put the model's assumptions to test, discover its limitations and

[5]Whether this constitutes an option depends on the situation. If any sort of randomization such as masking or re-keying is present in the device then the adversary is limited in the number of attack traces/samples that he can combine.

obtain more insight into the source of location leakage.

6.4.1 Region Partition

To observe the effect of partitioning, we gradually split the 256 bytes of the AES LUT into classes and built the corresponding template for each class. We perform a template attack on 2, 4, 8 and 16 partitions (with 128, 64, 32 and 16 bytes each respectively), i.e. we gauge the distinguishing capability of the adversary, as the number of components increases and their respective areas decrease. The results are visible in Figure 6.12, which showcases how the number of grid positions (spatial POIs) and time samples per attack affects the success rate (SR). The adversary can achieve



(a) 2 regions of 128 bytes each

(b) 4 regions of 64 bytes each

(c) 8 regions of 32 bytes each

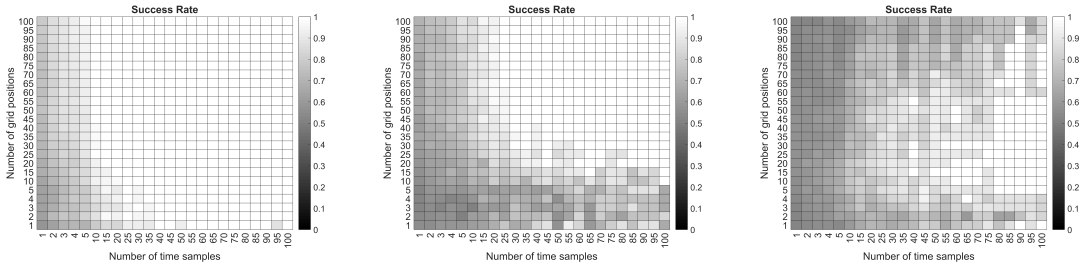
Figure 6.12: The success rate of the template-based classifier as we partition the AES LUT. Y-axis denotes the number of spatial POIs used in model, X-axis denotes the number of time samples used in attack. Scale denotes SR where white is 100% and black is 0%.

a success rate of 100% when distinguishing between 2 or 4 regions, assuming that he uses multiple time samples in his attack. The success rate drops to 75% for 8 and 50% for 16 regions, an improvement compared to random guess SRs of 12.5% and 6.25% respectively. Although we are not able to reach successful byte-level classification, we can safely conclude that location-based attacks are indeed possible on small LUTs and they can reduce the security level of an AES implementation, unless address randomization countermeasures are deployed. When performing *single-sample* attacks, the template strategy becomes less potent, achieving SR of 57%, 33%, 17% and 11% for 2, 4, 8 and 16 regions, i.e. only slightly better than a random guess. In order to compare the success rate of the real attack to the theoretical model, we compute the model's SR for current device SNR under the same data complexity[6]. The model's *single-sample* SR is 99%, 50%, 13% and 12% for 2, 4, 8 and 16 regions respectively. In this case, we observe that the model follows the same trend, yet the device leakage exhibits divergences that indicate modeling imperfections.

[6]The template attack uses the experimental data, while the theoretical SR uses simulated data of the same size and dimensionality.

6.4.2 Grid Dimension

Using the same approach, we evaluate the effect of grid dimension on the success rate of the template attack. We commence with the full 300×300 grid (2-day experiment) and subsequently scale down to 40×40 grid (1-hour experiment) and 10×10 grid (2-minute experiment). The results are visible in Figure 6.13. We observe that for



(a) 300×300 grid (b) 40×40 grid (c) 10×10 grid

Figure 6.13: The success rate of the 2-region template-based classifier as we decrease the experiment's grid size.

small grid sizes such as 10×10 the reduced dataset makes training harder, yet the multi-sample template attack is able to distinguish with SR equal to 100%. On the contrary, the theoretical model is unable classify correctly because the spatial POIs are often missed by such a coarse grid. To pinpoint this model limitation, we assess the spread of the POIs across the die surface and we visualize the best (according to correlation) grid positions in Figure 6.14. Interestingly, we discover numerous surface positions that leak location information, while being far away from the SRAM circuitry itself. This observation is in accordance with the findings of Unterstein et al [211] that observe various out-of-model localized leakages on FPGA devices. Overall, we speculate that location leakage is a combination of SRAM spatial leakage (as in the model) and other forms of leakage, stemming from different spatial components or even temporal features.

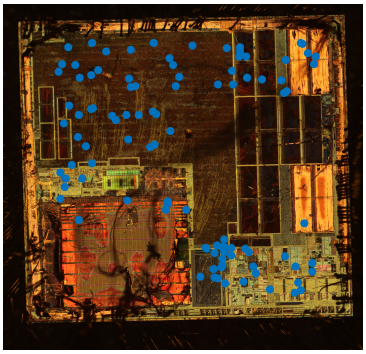
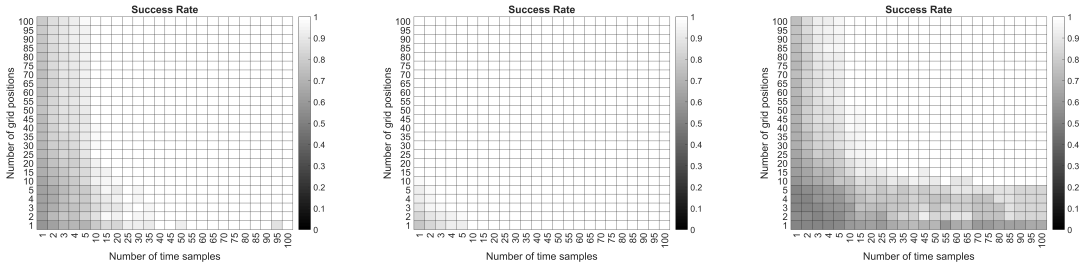


Figure 6.14: Spread of spatial POIs on chip surface.

6.4.3 Placement

Finally, we evaluate the effect of region proximity and interleaving on the SR of template attacks. We examine close placement (sequential SRAM regions), distant placement (SRAM regions at a large distance[7]) and word-interleaved placement (checkered SRAM regions). The results are visible in Figure 6.15.



(a) close (b) distant (c) word-interleaved

Figure 6.15: The success rate of the 2-region template-based classifier as we change the placement of regions.

We observe that in all cases we reach multi-sample SR of 100%, in accordance with the theoretical model at the device SNR. However, attacking the word-interleaved LUT requires a bigger effort in modeling in terms of both grid POIs and samples per attack. Likewise, distinguishing between distant regions puts a considerably less strain on the model. Thus, we conclude that distance and interleaving does indeed function like a countermeasure against location leakage, albeit it offers only mild protection in our ARM device.

6.5 Exploitation Using Neural Networks

Despite the fact that the multivariate normal leakage assumption is fairly realistic in the side channel context, applying distribution-agnostic techniques appears to be another rational approach [133]. Over the past few years, there has been a resurgence of interest in Deep Learning techniques, powered by the rapid hardware evolution and the need for rigorous SCA modeling [43, 138, 141, 142, 171, 222]. In this section, we evaluate the performance of convolutional neural networks (Subsection 6.5.1) and multi layer perceptrons (Subsection 6.5.2) in inferring the activated region of the AES LUT on the ARM Cortex-M4. Interestingly, certain NNs are able to surpass the template attacks in effectiveness, enabling stronger location-based attacks that use less time samples and can distinguish between smaller regions.

[7]Note that without knowledge of the chip layout we cannot be fully certain about the distance between memory addresses. Here we assume that the low addresses of the SRAM are sufficiently distant from mid ones, which are approx. 8 KBytes away.

6.5.1 Convolutional Neural Network Analysis

Before developing and customizing our own CNN model, we evaluate the performance of existing, state-of-the-art *fully pretrained CNNs*. Pre-trained models are usually large networks that have been trained for several weeks over vast image datasets. As a result, their first layers tend to learn very good, generic discriminative features. Transfer Learning [157] is a set of techniques that, given such a pre-trained network, repurposes its last few layers for another similar (but not necessarily identical) task. Indeed, the objectives of our spacial identification task appear to be very close to those of standard image classification. Moreover, as outlined in Section 6.2.1, our data is formulated as 300×300 grid images, which makes them compatible with the input format of several computer vision classification networks. For this first attempt at CNN classification we use several state-of-the-art networks, namely Oxford VGG16 and VGG19 [191], Microsoft ResNet50 [101], Google InceptionV3 [206] and Google InceptionResNetV2 [205]. It should be noted that the input format of these networks is often RGB images, while our 300×300 heatmaps resemble single-channel, grayscale images. To address this and recreate the three color channels that the original networks were trained for, we experiment with two techniques; (1) we assemble triplets of randomly chosen heatmaps, and (2) we recreate the three color channels by replicating the heatmaps of the samples three times.

We apply the pretrained CNN classification on 2 closely placed SRAM regions of 128 bytes each. In accordance with the standard transfer learning methodology, during re-training we freeze the first few layers of the networks to preserve the generic features they represent. In each re-training cycle, we perform several thousand training-testing iterations. Despite all these and multiple hours of training, none of the aforementioned CNNs results into a retrained network with high classification success rate. In all cases, the networks perform similarly to a random guess.

As a result of the low success rate of fully pretrained networks, we choose to proceed with Xavier [89] weight initialization and training from scratch, working towards *custom pretrained CNNs*. We observe that, despite the transformation of the sequential problem (SRAM accesses over time) to a spatial one, our dataset is dissimilar to visual classification datasets. Rephrasing, the images that we have to cope with feature intricate characteristics having little resemblance with those of the datasets that the pretrained CNN versions have been trained on, such as the ImageNet dataset [71]. Moreover, due to the fully distribution-agnostic approach, any randomly initialized CNN may suffer the effect of vanishing or exploding gradients, a danger that Xavier initialization should eliminate. The framework that was used for training and evaluating our customized CNNs is Keras [51] over TensorFlow [6] backend and the customized CNNs tested were VGG19 [191], InceptionV3 [206], ResNet50 [101], DenseNet121 [108] and Xception [52]. We also made use of the *scikit-learn* Python library [164] for the preprocessing of our data. The execution of this customized CNN training and testing was carried out in ARIS GRNET HPC (High-Performance Computing) infrastructure[8].

[8] `https://hpc.grnet.gr/en/`

To gauge the effect of SRAM memory addressing on the CNN training, all five CNNs are trained in two ways, namely one-batch training and multiple-batch training. During one-batch training we use location leakage from a single SRAM LUT, while for multiple-batch traing we use four LUTs placed within a 16 KByte SRAM address range. The dataset is split into training, validation and test sets using a 70-20-10 ratio and is standarized by removing the median and scaling the data according to the quantile range. The networks are trained for 150 epochs of 32 images each, using the Adam optimizer [120] with default parameters. The results are visible in Figure 6.16.

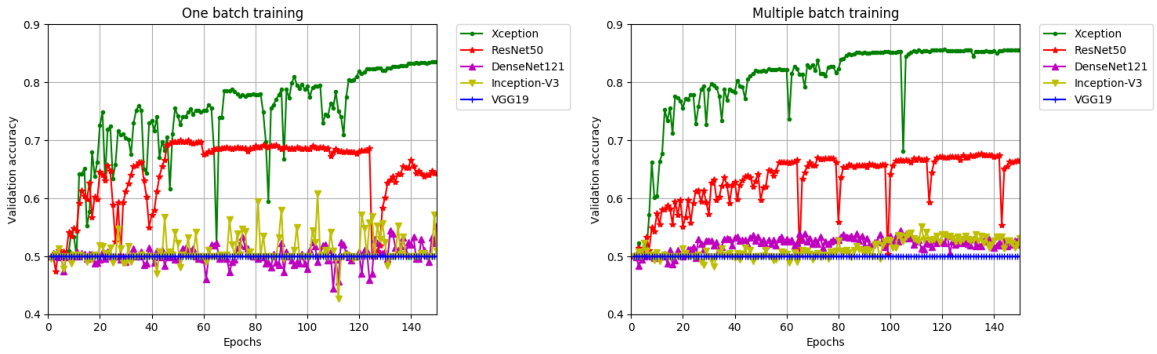
We observe that she single-sample success rate of the Xception network (green line) exceeds by far all others' at 84% and the SR improves in stability when using multiple-batch training. It is worth noting that some CNNs, especially VGG19, remain incapable of learning anything meaningful about the discrimination of the two 128-byte regions. Another troubling fact is the sudden drops of validation accuracy during training time for both best-performing networks, Xception and ResNet50, a phenomenon rather indicative of overfitting. In our efforts to squeeze the best possible performance without sacrificing training stability and generalization capacity, we investigated the tolerance of the best performing network against two additional preprocessing techniques, namely sample-wise standardization and feature-wise standardization. The test set success rate of the three alternative techniques is visible in Table 6.2. Comparing with Section 6.4, we observe that CNNs are capable of surpassing the single-sample accuracy of template attacks, reaching 88% and making the CNN-based attack particularly useful against randomization countermeasures that limit the number of samples we can combine. Moreover we observe that spreading the training phase over several SRAM addresses (multiple batch) can assist classification, showing that the knowledge learned in a certain address range may be applicable elsewhere in the SRAM.

Table 6.2: Success rate of Xception network for alternative preprocessing techniques.

Alternative Pipeline	**Preprocessing step**	**Success Rate**
Xception-V1	dataset-wise, robust to outliers standardization	84.47 %
Xception-V2	sample-wise standardization	88.636 %
Xception-V3	feature-wise standardization	84.848 %

6.5.2 Multi Layer Perceptron Network Analysis

In [138], the authors presented how to use Multi Layer Perceptron (MLP) network to perform SCA on AES. In this section we expand our NN-based analysis and we present how to use MLP to recognize accesses to different addresses in the memory. To improve learning results we employ dimensionality reduction techniques based on the correlation heuristic to detect the best spatial POIs in the 300×300 grid, similarly to template attacks in Section 6.4.1. Based on experiments, we discover that 5000 POIs yielded the best network training.



(a) Single-batch training.

(b) Multiple-batch training.

Figure 6.16: CNN validation accuracy for single/multiple-batch training.

We define our MLP to contain a single dense layer and used the back-propagation with NESTEROVS updater, with momentum 0.9, during training. The weights are initialized at random and applied to a RELU activation. The MLP is also configured with L1 and L2 regularization in order to improve the generalization. The detailed parameters are given in Table 6.3. The analysis presented in this section is performed using Riscure's Inspector software package[9], which is based on Deep Learning for Java[10]. To observe the effectiveness of MLPs, we gradually partition the 256 bytes of the AES LUT into classes and built the corresponding MLP for each class. We perform an MLP analysis on 2, 4, 8, 16, 32, 64, 128, and 256 partitions (with 128, 64, 32, 16, 8, 4, 2, and 1 bytes each, respectively). The dataset is split into training, validation and test sets using a 40-30-30 ratio. Then we select best hyper-parameters for training and validation[11] of our MLP network using a trial and error method. The chosen parameters are listed in Table 6.3.

The validation accuracy for 2, 4, 8, 16, 32, 64, 128, and 256 partitions is visible in Figure 6.17a. We have discovered that we achieve the best results for various numbers of epochs depending on the number of partitions. We have used 30 epochs for the 2 and 4 partitions, 40 epochs for the 16 and 32 partitions, 40 epochs for the 8 partitions, 70 for the 128 partitions, and 80 for the 64 and 256 partitions. Figure 6.17a indicates that the MLP network reaches high accuracy even for a large number of regions. To visualize the validation set success rate we present the validation final partitioning (for 16 partitions) in Table 6.4. The greatest values are located on the diagonal and this indicates that the MLP learns correctly with high probability. The attack success rates for the test traces for 2, 4, 8, 16, 32, 64, 128, and 256 partitions are presented in Figure 6.17b; the exact accuracy values are 96%, 91%, 90%, 88%, 83%, 75%, 57%, and 32%, respectively. As expected, these values are slightly lower for the attacking phase than the validation ones in the learning phase. We observe that even the SR for the 256 partitions, namely the 32% SR is significantly higher then a SR of a random

[9] `https://tinyurl.com/jlgfx95`

[10] `https://deeplearning4j.org/`

[11] The MLP parameters are chosen to maximize the attack success rate (which is equivalent to accuracy).

Epochs	$30 - 80$ (depends on the number of regions)
Mini-Batch	100
Learning Rate	0.003
Learning Rate Decay Rate	0.5%
Learning Rate Decay Interval	100 epochs
L1	0.001
L2 (weight decay)	0.001
Weight Initialization	RELU
Activation Output Layer	SOFTMAX
Loss Function	NEGATIVELOGLIKELIHOOD
Updater	NESTEROVS
1 Dense Layer:	
- Number of Neurons:	20
- Activation Dense Layer:	TANH

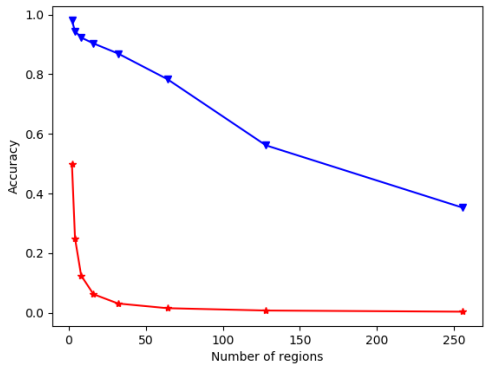
Table 6.3: Hyper-Parameters for training and validation.

Predicted:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Actual:																
0	35	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
1	1	30	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	40	1	1	0	0	0	0	0	0	0	0	0	0
4	2	0	0	1	31	0	0	1	0	0	0	0	1	0	0	0
5	0	1	0	1	0	28	1	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	37	0	0	0	0	0	0	1	0	0
7	0	0	0	1	2	1	0	26	0	0	0	0	0	0	1	0
8	0	0	0	1	0	0	0	0	26	0	0	0	0	0	0	0
9	0	0	1	0	0	1	1	0	1	30	0	0	0	0	0	0
10	0	0	0	0	0	2	0	0	1	1	37	0	1	0	0	2
11	0	1	0	1	1	0	0	1	0	1	0	34	0	1	0	0
12	0	1	0	0	0	0	0	2	1	0	0	0	34	0	0	0
13	0	0	0	0	0	0	1	1	0	0	3	0	0	32	2	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0
15	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	31

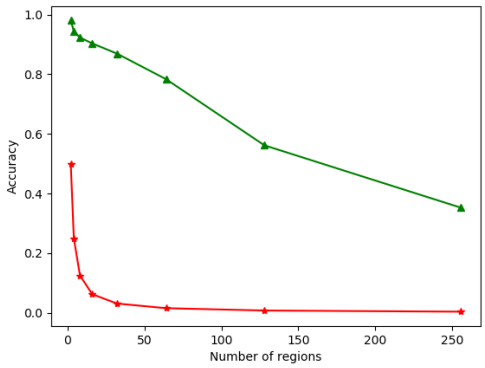
Table 6.4: Predicted versus actual values.

guess: $1/256 = 4\%$.

Observing these results, we conclude that the MLP network can be substantially stronger than the template attacks when exploiting location leakage. We note that it can achieve high SR using single-sample attacks, reaching 98% for 2 regions. Continuing, it can reach a SR of 32% even when targeting single bytes in the SRAM. Notably, the MLP classification can strongly enhance the SR of a microprobe setup making it almost on par with the substantially more expensive photonic emission setup. Still our comparison remains incomplete, i.e. it is unclear wheather LDA-based template attacks or some other POI configuration are able to match the MLP performance. An exact comparison between competing profiling/attacking SCA



(a) Blue line denotes validation accuracy and red line denotes random guess success rate.



(b) Green line denotes attack success rate and red line denotes random guess success rate.

Figure 6.17: Validation accuracy for training and success rate for testing in MLP.

techniques remains an open problem.

6.6 Linking Data and Location Leakage

This chapter identified two key issues with respect to zero-day hardware exploits: *modeling* the new exploit and *linking* it to existing attacks and countermeasures. Sections 6.2 until 6.5 focused on modeling, providing theoretical and practical attempts to describe the behavior of location-based leakage and gauge its impact on security. This section focuses on the linking the location leakage dimension to the existing data leakage dimension, aiming towards a quick integration of the novel exploit to our existing countermeasure infrastructure. Analytically, Section 6.6.1 describes the newly proposed Boolean Exponent Splitting technique from Tunstall et al. [209], a scheme that aims to mitigate both data-based and location-based leakage in the context of elliptic curve cryptography, by splitting an exponent in two Boolean shares. Continuing, Section 6.6.2 performs an information-theoretic analysis on the proposed scheme from the viewpoint of data-based leakage and location-based leakage. More importantly, it links these two types of leakage in a joint hybrid attack, thus *quantifying* the interaction between the novel exploit on location and the common exploit on data.

6.6.1 Boolean Exponent Splitting

The asymmetric cryptography literature has provided several techniques such as additive, multiplicative and Euclidean splitting [55, 57] that can prevent side-channel analysis during scalar multiplications. To the same end, Tunstall et al. [209] put forward several Boolean splitting methods, which can be applied to Montgomery ladder algorithms. In Algorithm 6 we recall the description of the Montgomery powering ladder given by Joye et al. [117], i.e. we consider the ladder for the scalar multiplication

$y = [k]x$, where x is an element of certain algebraic group $\mathbb{G}$, y is the public output of the algorithm and k is the n-bit secret key, denoted also as $k_{n-1} \dots k_0$.

Algorithm 4: Standard Montgomery Ladder

Input: $x \in \mathbb{G}$, n-bit integer $k = \sum_{i=0}^{n-1} k_i \, 2^i$
Output: $y = [k]x$

1 $R_0 \leftarrow 1_{\mathbb{G}}$; $R_1 \leftarrow x$;
2 **for** $i = n - 1$ ***down to*** 0 **do**
3 $\quad R_{\neg k_i} \leftarrow R_{k_i} \cdot R_{\neg k_i}$;
4 $\quad R_{k_i} \leftarrow (R_{k_i})^2$;
5 **end**
6 **return** R_0

The standard Montgomery ladder is highly regular, i.e. a deterministic sequence of operations is executed for an exponent of a given bit length, preventing straightforward timing attacks. Still, whenever the secret key k_i is manipulated by the device (loaded, stored or processed) it results in 1st-order data-based leakage that can compromise security. Similarly, register accesses such as R_{k_i} or $R_{\neg k_i}$ are key dependent operations, resulting in 1st-order location-based leakage.

To prevent 1st-order data-based and location-based leakage, Tunstall et al. [209], continuing the work of Izumi et al. [114], put forward Algorithm 8 which operates similarly to the MPL, yet also performs Boolean exponent splitting. The new algorithm splits the key k to shares a and b, each consisting of n bits and is proven secure against 1st-order attacks.

Algorithm 5: Boolean Exponent Split Montgomery Ladder

Input: $x \in \mathbb{G}$, n-bit integers $a = \sum_{i=0}^{n-1} a_i \, 2^i$ and $b = \sum_{i=0}^{n-1} b_i \, 2^i$
Output: $y = [k]x$, where $k = a \oplus b$

1 $R_0 \leftarrow 1_{\mathbb{G}}$; $R_1 \leftarrow 1_{\mathbb{G}}$; $U_0 \leftarrow x$; $U_1 \leftarrow x^{-1}$;
2 $b' \xleftarrow{\mathrm{R}} \{0, 1\}$; $R_{\neg b'} \leftarrow x$;
3 **for** $i = n - 1$ ***down to*** 0 **do**
4 $\quad R_0 \leftarrow R_{b_i \oplus b'} \cdot R_{(b_i \oplus b') \oplus a_i}$;
5 $\quad R_1 \leftarrow R_0 \cdot U_{b_i}$;
6 $\quad b' \leftarrow b_i$
7 **end**
8 **return** $R_{b'}$

6.6.2 Joint Information-Theoretic Evaluation

Having described a 1st-order probing-secure algorithm, we proceed to analyze the noise amplification stage of Boolean exponent splitting, in order to provide the full picture regarding the new countermeasure. We perform an evaluation of the Boolean exponent splitting (as described by Algorithm 8) using the information-theoretic framework of Standaert et al. [197]. Analogous approaches can be conducted for other variants of exponent splitting algorithms, yielding very similar results. Our analysis considers two sources of leakage, namely data-based leakage and location-based leakage (also known as address leakage in the literature). Using both leakage sources, we demonstrate three possible attack paths against Algorithm 8, covering multiple leakage combinations. Analytically, we show the noise amplification stage 1) when only data-based leakage is exploited (data attack), 2) when only location-based leakage is exploited (location attack) and finally the noise amplification stage 3) when the adversary combines data and location leakage (hybrid attack).

We begin by extending the current data-based notation to accommodate for location-based leakage. So far, observable data-based leakages of a certain intermediate value v are denoted using subscript L_v. We denote observable location-based leakages caused by accessing register R_j (where j the register index) using subscript $L_{R\text{-}j}$. To distinguish between data-based leakage and location-based leakage we use superscript L^{data} and L^{loc}. In addition, we assume that the two different sources of leakage (data, location) have different noise levels i.e. we assume homoscedastic data noise $N^{data} \sim \mathcal{N}(0, \sigma^2_{data})$ and homoscedastic location noise $N^{loc} \sim \mathcal{N}(0, \sigma^2_{loc})$. Subsequently, we use the following formula to compute the MI metric.

$$MI(S; \mathbf{L}^{tot}) = H[S] + \sum_{s \in \mathcal{S}} Pr[s] \cdot \sum_{m \in \{0,1\}} Pr[m] \cdot \int_{\mathbf{l}^{tot} \in \mathcal{L}^{\zeta}} Pr[\mathbf{l}^{tot}|s, m] \cdot log_2 Pr[s|\mathbf{l}^{tot}] \; d\mathbf{l}^{tot}$$

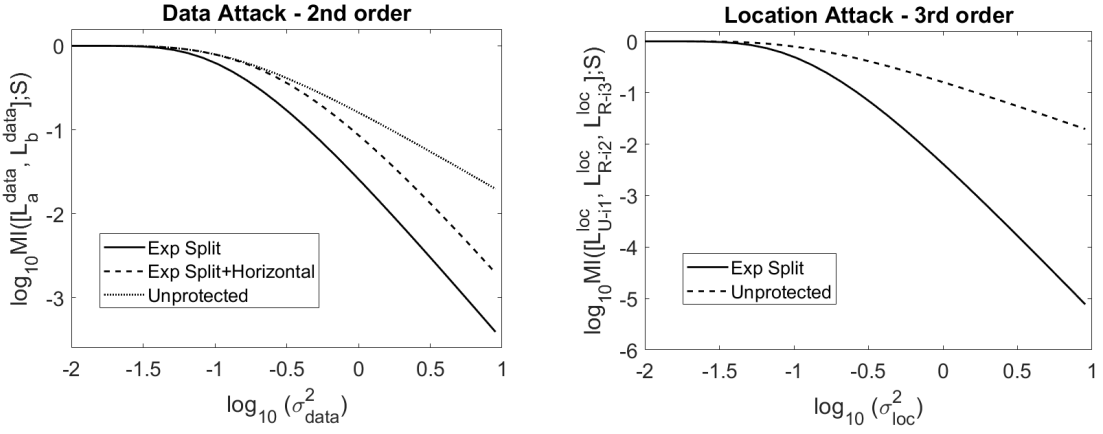
$$\text{where } Pr[s|\mathbf{l}^{tot}] = \frac{\sum_{m^* \in \{0,1\}} Pr[\mathbf{l}^{tot}|s, m^*]}{\sum_{s^* \in \mathcal{S}} \sum_{m^* \in \{0,1\}} Pr[\mathbf{l}^{tot}|s^*, m^*]}$$

Random variable S denotes the secret, i.e. an exponent key bit k_i. $\mathbf{L}^{tot}$ denotes the data-based or location-based leakage vector used in the evaluation and m is the random bit that we need to sum over and originates from the splitting of k_i to a_i and b_i. The leakage dimension ζ is adjusted according to the case (data-only, location-only, joint data-flocation) and to the amount of available horizontal leakage.

Data-based evaluation. The first obvious way to recover k_{n-1} is by observing the data leakage of the values b_{n-1} and a_{n-1} at the same time. We run the algorithm for the first two rounds and note the intermediate values that can leak information. We let $b' \in_R \{0, 1\}$, then for certain loop iteration m:

$i = n-1$	$i = n-2$
1. $b_m = b_{n-1} \oplus b'$	6. $b_m = b_{n-2} \oplus b'$
2. $a_m = b_m \oplus a_{n-1}$	7. $a_m = b_m \oplus a_{n-2}$
3. $R_0 = R_{b_m} \cdot R_{a_m}$	8. $R_0 = R_{b_m} \cdot R_{a_m}$
4. $R_1 = R_0 \cdot U_{b_{n-1}}$	9. $R_1 = R_0 \cdot U_{b_{n-2}}$
5. $b' = b_{n-1}$	10. $b' = b_{n-2}$

As can be observed in above, the value b_{n-1} is accessed in the first iteration ($i = n-1$) three times, once when b_m is calculated (line 1), once implicitly for the index of $U_{b_{n-1}}$ (line 4) and finally for b' (line 5). The value a_{n-1} is accessed once during the first iteration ($i = n-1$) and it is not used in the second iteration ($i = n-2$). We notice that the value b_{n-1} is used implicitly again in the second iteration, since it is equal to b'. An attacker observing the power leakage of this algorithm should be able to probe at two different points in time, in order to observe both leakages $L^{data}_{a_{n-1}}$, $L^{data}_{b_{n-1}}$ and eventually the key, i.e. we observe that a 2nd-order attack is possible for this scheme. Note also that the an adversary with ability to conduct horizontal side-channel attacks [25] could observe the leakage of b_{n-1} multiple times, average them by computing $\bar{L}^{data}_{b_{n-1}} = \frac{1}{4} * \sum_{j=1}^{4} L^{data}_{b_{n-1}}$ in order to reduce the noise level first and then perform a 2nd-order attack, much like our approach in Chapters 4 and 5. The results of the MI evaluation are visible in Figure 6.18a. As expected, the exponent splitting scheme performs noise amplification and has a different slope compared to an unprotected exponentiation (Algorithm 6) In addition, we observe the curve's horizontal shift to the right caused by the horizontal exploitation of the available leakage, i.e. we can quantify the effect of multiple leaky points for b_{n-1}.



(a) Data-based evaluation, w/wo horizontal exploitation. $\mathbf{L}^{tot} = [L^{data}_{a_{n-1}}, L^{data}_{b_{n-1}}]$.

(b) Location-based evaluation,using register indexes. $\mathbf{L}^{tot} = [L^{loc}_{U\text{-}i_1}, L^{loc}_{R\text{-}i_2}, L^{loc}_{R\text{-}i_3}]$.

Figure 6.18: Data-based and location-based MI evaluations for Algorithms 6, 8

Location-based evaluation. Let us assume that the adversary can distinguish between the manipulation of registers according to which address is accessed, similar

to the address-bit DPA attack described in [114]. If the adversary can sufficiently distinguish between accesses to U_0 and U_1 for example, a direct consequence is recovery of value b_{n-1}. To mount a successful attack against Algorithm 8 using solely location-based leakage, we need the simultaneous observation of the address of U_{i_1} and R_{i_2} and R_{i_3}, for indexes $i_1 = b_{n-1}$ (line 4) and $i_2 = b_m$ (line 3) and $i_3 = a_m$ (line 3). Thus, in order to recover k_{n-1}, we need to observe leakage vector $L^{loc} = [L^{loc}_{U\text{-}i_1}, L^{loc}_{R\text{-}i_2}, L^{loc}_{R\text{-}i_3}]$, i.e. perform a 3rd-order attack. The results are visible in Figure 6.18b, where we can observe the noise amplification effect that increases the curve's slope. Naturally, if $\sigma_{loc} = \sigma_{data}$, a 3rd-order attack using only location-based leakage will be less effective compared to a 2nd-order attack using only data-based leakage. However, depending on the device, exploiting the address dependency may be more effective than exploiting the data dependency. That is, the 3rd-order attack can become more efficient if $\sigma_{data} > \sigma_{loc}$, showing that ignoring location-based leakage can prove dangerous.

Hybrid leakage attack. Finally, we analyze the scenario in which an adversary can observe both data-based and location-based leakage. Using this information the adversary can use leakage vector $\mathbf{L} = [L^{data}_{a_{n-1}}, L^{loc}_{U\text{-}j}], j = b_{n-1}$ to carry out a 2nd-order attack that uses data leakage to recover bit a_{n-1} and location leakage from register U to recover bit b_{n-1}. Since data and location leakage imply different noise levels ($\sigma_{data} \neq \sigma_{loc}$), we need to represent the available information as a three-dimensional plot, as in Figure 6.19. The wave-like plot quantifies the attainable information with regard to a particular data and location noise level. Thus, it assists the side-channel evaluator to analyze the scheme's security in a more holistic way compared to Figure 6.18a and 6.18b. Factoring in both data and location leakage demonstrates the tradeoff between data noise and location noise. If for instance $\sigma_{loc} \ll \sigma_{data}$ in the target device, the adversary can directly opt for the hybrid attack, instead of pursing a data-only attack route. Moreover, linking the more recent location-based exploit to existing attacks enables customized countermeasure design, where the designer can maximize protection by adapting to the device-specific noise levels.

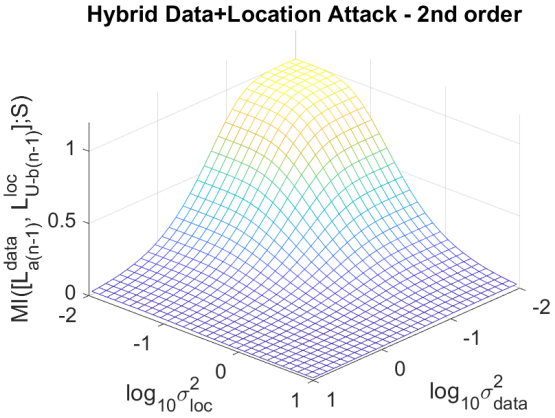


Figure 6.19: MI evaluation for Algorithm 5 exponent splitting, using a hybrid leakage attack. Observed leakage vector $\mathbf{L} = [L^{data}_{a_{n-1}}, L^{loc}_{U\text{-}b_{n-1}}]$.

6.7 Conclusions & Future Directions

In this chapter, we have revisited the potent, yet often overlooked location-based leakage. We take the first steps towards theoretical modeling of such effects and we put forward a simple spatial model to capture them. Continuing, we demonstrate successful location-based attacks on a modern ARM Cortex-M4 using both standard template attacks, CNNs and MLPs. Throughout these attacks we assess the impact of various experimental parameters, as well as interactions with common data-based SCA, in order to elucidate the nature and exploitability of location-based leakage. Reconsidering the impact of zero-day vulnerabilities (such as location-based leakage), we suggest that although such vulnerabilities are persistent in the security field, it is possible and highly recommended to quickly integrate any new findings into existing frameworks in order to quickly mitigate their impact and provide holistic protection against them.

Regarding future work, we note that during the last years of side-channel research, the community has established a multitude of potent tools (ranging from Bayesian techniques to neural networks), all of which are particularly good at extracting the available leakage. Still, we remain far less capable of finding the exact cause behind it, especially in complex modern chips [12, 161]. Thus, a natural extension to this work is to delve deeper into the electrical layer of a system-on-chip, try to identify the "culprint" behind location leakage and ultimately diminish the emitted information. In the same spirit, we should strive towards improved circuit modeling, similarly to the works of Šijačić et al. [189] and Kumar et al. [129], adapt them to the spatial model and use it in order to shorten the development-testing cycle of products.

Once again, this chapter observes a similar trend to Chapters 4 and 5, regarding countermeasure interactions. The chapter has already analyzed masking schemes that account for data *and* location leakage. Rephrasing, the location channel is yet another parameter under consideration by the countermeasure designer. Naturally, the new parameter is directly linked to existing attacks, prompting an joint analysis. In this direction, we can continue towards higher-order data *and* location masking schemes in order to provide fully-fledged security that encapsulates multiple side-channel vulnerabilities.

Chapter 7

The Next Decade in Side-Channels

"Because things are the way they are, things will not stay the way they are."

Bertolt Brecht, 1976

7.1 Conclusions

This thesis has shed light on multiple aspects of side-channel analysis, ranging from masked cryptographic implementations to fault-resistant designs and location-based attacks. Below we sum up the core conclusions of this thesis, as they appeared through its chapters.

- Chapter 3 has shown that higher-order masking is not simply a theoretical construct but it can be hosted in most modern microcontrolers. Our ability to efficiently code protected cipher provides the various IoT devices with affordable, yet robust protection, thus can usher us into a new era of safe pervasive computing.
- Chapters 4 and 5 have shown the close relationship and subtle interactions between SCA, FI and RNG. In this process we identified numerous interactions that resemble a "doulbe-edged sword": dangerous for security if left unchecked, yet potent tradeoffs if used by the designer. This detailed analysis of interactions has led to more efficient and holistically protected cryptographic designs.
- Chapter 6 has shown the impact of location-based side-channel attacks, while integrating them to existing protection mechanisms. Furthermore, the chapter has showcased the effectiveness of deep learning techniques and justified their increasing popularity in hardware security applications.

7.2 Future Directions

In this thesis, we have also identified several persisting problems such as the large gap between theory and practice in hardware security, the limited understanding

of the nature of side-channel leakages and the increasingly large amount of design options. This section, attempts a future sight, that is, it discusses several topics that we deem very likely to affect the side-channel community in the coming decade. Looking towards the future of side-channel analysis, we focus on three core topics that we consider both important and challenging enough to prompt theoretical and applied research. To do so, we shift towards a more critical stance against the work carried out in this thesis and the hardware community in general. Our aim is not to dismiss our existing work due to its imperfections, since this work led us here in the first place. Instead, we aim to highlight current limitations and show how future research can surpass them.

7.2.1 Searching for optimal attacks and vulnerabilities

A persisting issue of side-channel analysis is the effective exploitation of leakage in a given dataset. The research effort in SCA has provided the community with a plethora of tools, starting from classical statistics, moving to statistical machine learning (template attacks, probabilistic graphic models) and ultimately to deep learning techniques. Chapter 6 confirms in practice the potential of novel deep learning techniques and we expect such research trends to continue in the coming years.

However, this vast array of attack tools requires a more generic attack strategy. As our arsenal keeps increasing, we need to identify which tools perform best, how can they be combined and whether they are able to adapt to the changing hardware environment. Thus, we maintain that a very promising goal for future research is to closely investigate learning tools in the context of leakage exploitation, but also in the context of knowledge transferability and hyper-heuristic search.

Moreover, we are recently detecting a rapidly widening gap between attack tools and defense mechanisms. The hardware security community in general has become increasingly capable of exploiting vulnerabilities in the SCA and FI context. For instance, Chapter 6 has deployed deep learning techniques and substantially improved the success rate of the SCA attack. Similarly, the improved search strategies of Carpi et al. [48] signify faster FI attacks. However, such advanced techniques are detached from the hardware vulnerability that causes them. Rephrasing, an evaluator that succeeded in "1st-order univariate CPA with HW model on the Sbox output" has a fairly clear understanding of the underlying issue and could work towards mitigating it. On the contrary, an evaluator that succeeded in a multi-layered neural network attack has limited ability to pinpoint and fix the root cause. Hence, we project that this attack-defense gap will increase in the coming years and we will need to balance our exploitation with respect to our ability of understanding it.

7.2.2 Increasing speculation and the physical layer

In most thesis chapters we have pinpointed several physical phenomena with a large impact on side-channel security. In Chapter 3, the large gap between theory and practice for masking schemes can be largely attributed to the discrepancy between

our theoretical assumptions (independent leakage assumption) and the phenomena exhibited at the electrical layer (distance-based leakages, coupling, memory effects, etc.). Unfortunately, instead of closely examining the electrical phenomena, the chapter resorts to speculative arguments about their nature, often motivated by additional experiments. Although the speculative arguments are thoroughly logical, they are still very easy to falter, since they lack an in-depth view of the underlying physical cause. A similar argument can be made for Chapter 6. The chapter puts forward a theoretical model for location-based leakage, after close inspection of the physical phenomenon. The spatial features suggested by the model are valid and backed by experimental results. Still, the high complexity of the electrical layer in a SoC makes the model stumble, since it relies on a fairly high-level view of the issue.

This increasing amount of speculation about the physical layer is also haunting the rest of the side-channel community. A prime example is the novel work of De Cnudde et al. [58] on coupling. The work follows a solid reasoning approach that adds progressively countermeasures. Eliminating leakages step-by-step leads the work towards *out-of-model* leakages, which in turn can be partially explained by coupling effects. Still, this elimination process cannot provide an in-depth explanation of low-level effects, prompting speculative discussions. Oddly, the increasing speculation about physical phenomena bears a close resemblance to natural sciences, where the common approach is to first formulate a hypothesis and then conduct laboratory experiments to prove or disprove it. However, this approach in physics, chemistry or biology is largely motivated by our limited understanding of highly complex structures, such as elusive energy particles, cells or even living organisms. Unlike these structures, any device-under-test in SCA is a well-studied artificial construct that can be understood and simulated to a reasonable extent. Since this is the case, the community should strive more towards precise modeling and experimentation and rely less on speculative discussions. In addition, moving the physical layer's security requirements to higher abstraction layers can also be problematic. For instance, it may be possible to construct high-level coupling-resistant and glitch-resistant masking schemes in the robust probing model of Faust et al. [84], yet a limited understanding of coupling and glitches can lead to overestimating the attacker's capabilities impose a heavy computational burden in the abstract model and hence the protected implementation.

To limit this hazardous speculation, we repeat the necessity for in-depth electrical layer research. The main "culprit" is the lack of transparency: side-channel evaluators are not often chip manufacturers, thus they have limited, if any, access to the physical layer of a device-under-test. Consequently, the community needs to strive towards more open designs where researchers can isolate specific effects, which in turn will enhance abstract models (such as the robust probing model) with the necessary information.

7.2.3 Automating interactive and parametric protection

We have seen that after identifying a hardware vulnerability, the standard approach is a three-stage process of designing a suitable countermeasure against it, implementing it in software/hardware and finally evaluating it a laboratory setting. The

direct consequence of this process is that hardware security components such as SCA countermeasures, FI countermeasures, RNG and the underlying physical layer are often analyzed and evaluated *in isolation* from each other. In numerous ocasions, this isolation can weaken protection, lure the device evaluator into a false sense of security or result in suboptimal devices. Chapter 5 has examined such interactions between previously isolated SCA and FI countermeasures, demonstrating how various fault injection countermeasures can reveal additional side-channel information to the adversary. Likewise, Chapter 4 investigated the impact of RNG on popular countermeasures like masking and shuffling. The exploration of these isolated components led to randomness recycling, a technique that offers effective protection, while keeping the cost affordable. In the same manner, the location-based vulnerabilities of Chapter 6 led to new countermeasures that must not be isolated and should be linked it to existing protection mechanisms.

Overall, reviewing the state of hardware security reveals many aspects that are examined in isolation. We observe that, unfortunately, the whole is more than the sum of its parts, since unforeseen interactions can hinder the development of secure devices. The design of a secure device often dissolves into isolated attempts at addressing various vulnerabilities such as SCA, location-based SCA, FI and others. Although targeted countermeasures can guarantee the effective protection of a problematic component, they largely ignore possible hazardous interactions. Consequently, hardware security is often trapped in a vicious cycle of discovering and securing vulnerabilities, while neglecting the function creep introduced by new countermeasures.

As a result, we maintain that future work must strive towards interactive and parametric design for countermeasures. Ideally, this new design paradigm should replace isolated countermeasures with a process that integrates a priori the multiple (and often conflicting) security requirements during the design phase of a product. Instead of viewing a countermeasures as standalone components, we opt to view them as an integrated system where they interact between each other and thus the designer needs to strike the correct balance. Working towards robust models that integrate the majority of countermeasures can result in a unified treatment and offer numerous combinations of countermeasures to choose from. Such plethora of choices calls for an automated selection process that will select the countermeasure parameters that maximize protection, minimize cost and help us move from standard design processes to a more tweakable automated parametric approach.

Appendix A

Author's Publication List

Thesis publications

- Bitsliced Masking and ARM: Friends or Foes?, *Wouter de Groot, Kostas Papagiannopoulos, Antonio de la Piedra, Erik Schneider and Lejla Batina*, Lightweight Cryptography for Security and Privacy - 5th International Workshop, LightSec 2016
- Vectorizing Higher-Order Masking , *Benjamin Grégoire, Kostas Papagiannopoulos Peter Schwabe and Ko Stoffelen*, Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018
- Mind the Gap: Towards Secure 1st-Order Masking in Software, *Kostas Papagiannopoulos and Nikita Veshchikov*, Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017
- Low Randomness Masking and Shuffling: An Evaluation Using Mutual Information IACR Transactions on Cryptographic Hardware and Embedded Systems, *Kostas Papagiannopoulos*, Volume 3, 2018
- Towards Lightweight Cryptographic Primitives with Built-in Fault-Detection, *Thierry Simon, Lejla Batina, Joan Daemen, Vincent Grosso, Pedro Maat Costa Massolino, Kostas Papagiannopoulos, Francesco Regazzoni, Niels Samwel*, to appear in Eurocrypt 2020
- Instruction Duplication: Leaky and Not Too Fault-Tolerant!, *Lucian Cojocar, Kostas Papagiannopoulos and Niek Timmers*, Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017
- Location, location, location: Revisiting Modeling and Exploitation for location-based side channel leakages, *Christos Andrikos, Lejla Batina, Lukasz Chmielewski, Liran Lerman, Vasilios Mavroudis, Kostas Papagiannopoulos, Guilherme Perin, Giorgos Rassias and Alberto Sonnino*, 25th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology ASIACRYPT 2019

- Location-based leakages: New directions in modeling and exploiting, *Christos Andrikos, Giorgos Rassias, Liran Lerman, Kostas Papagiannopoulos and Lejla Batina*, International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS 2017.

- Boolean Exponent Splitting, *Mike Tunstall, Louiza Papachristodoulou and Kostas Papagiannopoulos*, under submission

Related publications

- Speed and Size-Optimized Implementations of the PRESENT Cipher for Tiny AVR Devices, *Konstantinos Papagiannopoulos and Aram Verstegen*, Radio Frequency Identification - Security and Privacy Issues 9th International Workshop, RFIDsec 2013

- High Throughput in Slices: The Case of PRESENT, PRINCE and KATAN64 Ciphers, *Kostas Papapagiannopoulos*, Radio Frequency Identification: Security and Privacy Issues - 10th International Workshop, RFIDSec 2014

- How Old Is Your Brain? Slow-Wave Activity in Non-rapid-eye-movement Sleep as a Marker of Brain Rejuvenation After Long-Term Exercise in Mice, *Maria-Panagiotou, Kostas Papagiannopoulos, Jos Rohling, Johanna Meijer and Tom Deboer*, Frontiers in Aging Neuroscience, Volume 10, 2018

- Confused by Confusion: Systematic Evaluation of DPA Resistance of Various S-boxes, *Stjepan Picek, Kostas Papagiannopoulos, Barış Ege, Lejla Batina, and Domagoj Jakobovic*, Progress in Cryptology - INDOCRYPT 2014"

- Optimality and beyond: The case of 4x4 S-boxes, *Stjepan Picek, Barış Ege, Kostas Papagiannopoulos, Lejla Batina, and Domagoj Jakobovic*, IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) 2014

- Side-Channel Based Intrusion Detection for Industrial Control Systems, *Pol Van Aubel, Kostas Papagiannopoulos, Lukasz Chmielewski and Christian Doerr*, Critical Information Infrastructures Security - 12th International Conference, CRITIS 2017

Appendix B

Research Data Management

This thesis research has been carried out under the institute research data management policy of Radboud University, Institute for Computing and Information Sciences. The policy can be found here:

`https://www.ru.nl/icis/research-data-management/policy-protocol/`
last date accessed: 1-4-2020.

An index of the side-channel measurement datasets used in this thesis and their respective location can be found here:

`https://gitlab.science.ru.nl/kpapagiannopoulo/kostas_thesis`

Bibliography

[1] Ceasar cipher. https://en.wikipedia.org/wiki/Caesar_cipher.

[2] Da vinci: Father of cryptography? https://www.wired.com/2003/04/da-vinci-father-of-cryptography/.

[3] Enigma machine. https://en.wikipedia.org/wiki/Enigma_machine.

[4] Mifare hack. https://www.youtube.com/watch?v=NW3RGbQTLhE.

[5] Rijndaelfast implementation on atmel avr. http://point-at-infinity.org/avraes/.

[6] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[7] Christos Andrikos, Lejla Batina, Lukasz Chmielewski, Liran Lerman, Vasilios Mavroudis, Kostas Papagiannopoulos, Guilherme Perin, Giorgos Rassias, and Alberto Sonnino. Location, location, location: Revisiting modeling and exploitation for location-based side channel leakages. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 285–314. Springer, 2019.

[8] Christos Andrikos, Giorgos Rassias, Liran Lerman, Kostas Papagiannopoulos, and Lejla Batina. Location-based leakages: New directions in modeling and exploiting. In *2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pages 246–252, July 2017.

[9] 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*, pages 1–14, 2006.

[10] Pol Van Aubel, Kostas Papagiannopoulos, Lukasz Chmielewski, and Christian Doerr. Side-channel based intrusion detection for industrial control systems. In *Critical Information Infrastructures Security - 12th International Conference, CRITIS 2017, Lucca, Italy, October 8-13, 2017, Revised Selected Papers*, pages 207–224, 2017.

[11] Josep Balasch, Sebastian Faust, and Benedikt Gierlichs. Inner product masking revisited. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 486–510, 2015.

[12] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, pages 64–81, 2014.

[13] Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. DPA, bitslicing and masking at 1 GHz. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems — CHES 2015*, volume 9293, pages 599–619, 2015. `http://eprint.iacr.org/2015/727.pdf`.

[14] Josep Balasch, Benedikt Gierlichs, Roel Verdult, Lejla Batina, and Ingrid Verbauwhede. Power analysis of atmel cryptomemory - recovering keys from secure eeproms. 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*, pages 19–34, 2012.

[15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 411–436, 2015.

[16] Alessandro Barenghi, Luca Breveglieri, Israel Koren, Gerardo Pelosi, and Francesco Regazzoni. Countermeasures against fault attacks on software implemented AES. page 7, 01 2010.

[17] Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators. `http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf`.

[18] Thierno Barry, Damien Couroussé, and Bruno Robisson. Compilation of a countermeasure against instruction-skip fault attacks. In *Proceedings of the Third Workshop on Cryptography and Security in Computing Systems*, CS2 '16, pages 1–6, New York, NY, USA, 2016. ACM.

[19] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 116–129. ACM, 2016. `http://eprint.iacr.org/2015/506.pdf`.

[20] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified proofs of higher-order masking. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology — EUROCRYPT 2015*, volume 9056, pages 457–485, 2015. `https://eprint.iacr.org/2015/060.pdf`.

[21] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology — EUROCRYPT 2017*, volume 10210, pages 535–566, 2017. `http://eprint.iacr.org/2016/912.pdf`.

[22] Lejla Batina, Łukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. Online template attacks. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology – INDOCRYPT 2014*, pages 21–36, Cham, 2014. Springer International Publishing.

[23] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *J. Cryptology*, 24(2):269–291, 2011.

[24] Lejla Batina, Jip Hogenboom, and Jasper G. J. van Woudenberg. Getting more from pca: First results of using principal component analysis for extensive power analysis. In Orr Dunkelman, editor, *Topics in Cryptology – CT-RSA 2012*, pages 383–397, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[25] Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 23–39, 2016.

[26] Alberto Battistello and Christophe Giraud. A note on the security of CHES 2014 symmetric infective countermeasure. In *COSADE '16*, pages 144–159, 2016.

[27] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. *IACR Cryptology ePrint Archive*, 2016:660, 2016.

[28] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 616–648, 2016.

[29] Daniel J. Bernstein. Cache-timing attacks on AES. `https://cr.yp.to/antiforgery/cachetiming-20050414.pdf`, 2005.

[30] Daniel J. Bernstein and Peter Schwabe. NEON crypto. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems — CHES 2012*, volume 7428, pages 320–339, 2012. `https://cr.yp.to/highspeed/neoncrypto-20120320.pdf`.

[31] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. A low-entropy first-degree secure provable masking scheme for resource-constrained devices. In *Proceedings of the Workshop on Embedded Systems Security*, WESS '13, pages 7:1–7:10, New York, NY, USA, 2013. ACM.

[32] Eli Biham. A fast new DES implementation in software. In *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, pages 260–272, 1997.

[33] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, Jan 1991.

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

[35] Alex Biryukov and David Wagner. Slide attacks. In Lars Knudsen, editor, *Fast Software Encryption*, pages 245–259, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[36] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, pages 450–466, 2007.

[37] Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of checking cryptographic protocols for faults. In *International conference on the theory and applications of cryptographic techniques*, pages 37–51. Springer, 1997.

[38] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. 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*, pages 208–225, 2012.

[39] Paul Bottinelli and Joppe W. Bos. Computational aspects of correlation power analysis. *IACR Cryptology ePrint Archive*, 2015:260, 2015.

[40] Joan Boyar and René Peralta. A new combinational logic minimization technique with applications to cryptology. In *Experimental Algorithms, 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings*, pages 178–189, 2010.

[41] 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*, pages 16–29, 2004.

[42] David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[43] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 45–68, 2017.

[44] Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 272–288, 2009.

[45] D. Canright and Lejla Batina. A very compact "perfectly masked" s-box for AES. In *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings*, pages 446–459, 2008.

[46] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-order masking schemes for s-boxes. In *Fast Software*

Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers, pages 366–384, 2012.

[47] Claude Carlet, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Algebraic decomposition for probing security. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 742–763, 2015.

[48] Rafael Boix Carpi, Stjepan Picek, Lejla Batina, Federico Menarini, Domagoj Jakobovic, and Marin Golub. Glitch it if you can: Parameter search strategies for successful fault injection. In *CARDIS*, volume 8419 of *Lecture Notes in Computer Science*, pages 236–252. Springer, 2013.

[49] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666, pages 398–412, 1999. `http://dx.doi.org/10.1007/3-540-48405-1_26`.

[50] 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*, pages 13–28, 2002.

[51] François Chollet et al. Keras. `https://keras.io`, 2015.

[52] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[53] M. O. Choudary and M. G. Kuhn. Efficient, portable template attacks. *IEEE Transactions on Information Forensics and Security*, 13(2):490–501, Feb 2018.

[54] 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*, pages 253–270, 2013.

[55] Matthieu Ciet and Marc Joye. (virtually) free randomization techniques for elliptic curve cryptography. In S. Qing, D. Gollmann, and J. Zhou, editors, *ICICS 2003*, volume 2836 of *LNCS*, pages 348–359. Springer, 2003.

[56] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In Miguel Soriano, Sihan Qing, and Javier López, editors, *Information and Communications Security*, pages 46–61, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[57] Christophe Clavier and Marc Joye. Universal exponentiation algorithm. In C. K. Koç, D. Naccache, and C. Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 300–308. Springer, 2001.

[58] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, pages 1–18, 2017.

[59] Lucian Cojocar, Kostas Papagiannopoulos, and Niek Timmers. Instruction duplication: Leaky and not too fault-tolerant! In *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, pages 160–179, 2017.

[60] Jeremy Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, and Pankaj Rohatgi. Test vector leakage assessment (TVLA) methodology in practice, 2013. `http://icmc-2013.org/wp/wp-content/uploads/2013/09/goodwillkenworthtestvector.pdf`.

[61] Jean-Sébastien Coron. Higher order masking of look-up tables. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 441–458, 2014.

[62] Jean-Sebastien Coron. Formal verification of side-channel countermeasures via elementary circuit transformations. Cryptology ePrint Archive, Report 2017/879, 2017. `https://eprint.iacr.org/2017/879`.

[63] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, pages 69–81, 2012.

[64] Nicolas Courtois, Daniel Hulme, and Theodosis Mourouzis. Solving circuit optimisation problems in cryptography and cryptanalysis. *IACR Cryptology ePrint Archive*, 2011:475, 2011.

[65] Joan Daemen. Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 137–153, 2017.

[66] Joan Daemen, René Govaerts, and Joos Vandewalle. A new approach to block cipher design. In *Fast Software Encryption, Cambridge Security Workshop, Cambridge, UK, December 9-11, 1993, Proceedings*, pages 18–32, 1993.

[67] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher square. In *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, pages 149–165, 1997.

[68] Joan Daemen, Michael Peeters, and Gilles Van Assche. Bitslice ciphers and power analysis attacks. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, pages 134–149, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[69] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[70] Wouter de Groot, Kostas Papagiannopoulos, Antonio de la Piedra, Erik Schneider, and Lejla Batina. Bitsliced masking and ARM: friends or foes? In *Lightweight Cryptography for Security and Privacy - 5th International Workshop, LightSec 2016, Aksaray, Turkey, September 21-22, 2016, Revised Selected Papers*, pages 91–109, 2016.

[71] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[72] A. Adam Ding, Cong Chen, and Thomas Eisenbarth. Simpler, faster, and more robust t-test based leakage detection. *IACR Cryptology ePrint Archive*, 2015:1215, 2015.

[73] Daniel Dinu, Yann Le Corre, Dmitry Khovratovich, Léo Perrin, Johann Großschädl, and Alex Biryukov. Triathlon of lightweight block ciphers for the internet of things. *NIST Lightweight Cryptography Workshop 2015*, 2015:209, 2015.

[74] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: exploiting ineffective fault inductions on symmetric cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 547–572, 2018.

[75] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptographic Engineering*, 1(2):123–144, 2011.

[76] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete — Or how to evaluate the security of any leaking device. In *Advances in Cryptology — EUROCRYPT 2015*, volume 9056, pages 401–429, 2015. `https://eprint.iacr.org/2015/119.pdf`.

[77] François Durvaux, Mathieu Renauld, François-Xavier Standaert, Loïc van Oldeneel tot Oldenzeel, and Nicolas Veyrat-Charvillon. Efficient removal of random delays from embedded software implementations using hidden markov models. In *CARDIS '12*, pages 123–140, 2012.

[78] François Durvaux, François-Xavier Standaert, and Santos Merino Del Pozo. Towards easy leakage certification. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems — CHES 2016*, volume 9813, pages 40–60, 2016. `https://eprint.iacr.org/2015/537.pdf`.

[79] François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. How to certify the leakage of a chip? In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology — EUROCRYPT 2014*, volume 8441, pages 459–476, 2014. http://eprint.iacr.org/2013/706.pdf.

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

[81] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. CRC press, 1994.

[82] Thomas Eisenbarth, Zheng Gong, Tim Güneysu, Stefan Heyse, Sebastiaan Indesteege, Stéphanie Kerckhof, François Koeune, Tomislav Nad, Thomas Plos, Francesco Regazzoni, François-Xavier Standaert, and Loïc van Oldeneel tot Oldenzeel. Compact implementation and performance evaluation of block ciphers in ATtiny devices. In *Progress in Cryptology - AFRICACRYPT 2012 - 5th International Conference on Cryptology in Africa, Ifrance, Morocco, July 10-12, 2012. Proceedings*, pages 172–187, 2012.

[83] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. On the power of power analysis in the real world: A complete break of the keeloqcode hopping scheme. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 203–220, 2008.

[84] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 89–120, 2018.

[85] Sebastian Faust, Clara Paglialonga, and Tobias Schneider. Amortizing randomness complexity in private circuits. Cryptology ePrint Archive, Report 2017/869, 2017. http://eprint.iacr.org/2017/869.

[86] Yunsi Fei, Qiasi Luo, and A. Adam Ding. A statistical model for DPA with novel algorithmic confusion analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, pages 233–250, 2012.

[87] Pierre-Alain Fouque and Frederic Valette. The doubling attack – why upwards is better than downwards. In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, pages 269–280, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[88] Benedikt Gierlichs, Jörn-Marc Schmidt, and Michael Tunstall. Infective computation and dummy rounds: Fault protection for block ciphers without check-before-output. In *Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings*, pages 305–321, 2012.

[89] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9, pages 249–256, May 2010.

[90] 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*, pages 158–172, 1999.

[91] Dahmun Goudarzi and Matthieu Rivain. On the multiplicative complexity of boolean functions and bitsliced higher-order masking. *IACR Cryptology ePrint Archive*, 2016:557, 2016.

[92] Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology — EUROCRYPT 2017*, volume 10210, pages 567–597, 2017.

[93] Joey Green, Arnab Roy, and Elisabeth Oswald. A systematic study of the impact of graphical models on inference-based attacks on AES. *IACR Cryptology ePrint Archive*, 2018:671, 2018.

[94] Benjamin Grégoire, Kostas Papagiannopoulos, Peter Schwabe, and Ko Stoffelen. Vectorizing higher-order masking. In *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings*, pages 23–43, 2018.

[95] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-designs: Bitslice encryption for efficient masked software implementations. In *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, pages 18–37, 2014.

[96] Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. Masking vs. multiparty computation: how large is the gap for AES? *J. Cryptographic Engineering*, 4(1):47–57, 2014.

[97] Vincent Grosso, François-Xavier Standaert, and Emmanuel Prouff. Low entropy masking schemes, revisited. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, pages 33–43, 2013.

[98] Vincent Grosso and François-Xavier Standaert. Masking proofs are tight (and how to exploit it in security evaluations). Cryptology ePrint Archive, Report 2017/116, 2017. http://eprint.iacr.org/2017/116.

[99] Sylvain Guilley, Laurent Sauvage, Florent Flament, Vinh-Nga Vong, Philippe Hoogvorst, and Renaud Pacalet. Evaluation of power constant dual-rail logics countermeasures against DPA with design time security metrics. *IEEE Transactions on Computers*, 59(9):1250–1263, 2010.

[100] Qian Guo, Vincent Grosso, and François-Xavier Standaert. Modeling soft analytical side-channel attacks from a coding theory viewpoint. *IACR Cryptology ePrint Archive*, 2018:498, 2018.

[101] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[102] Wei He, Eduardo de la Torre, and Teresa Riesgo. An interleaved EPE-immune PA-DPL structure for resisting concentrated EM side channel attacks on fpga implementation. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 39–53, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[103] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, pages 239–252, 2006.

[104] Johann Heyszl. Impact of Localized Electromagnetic Field Measurements on Implementations of Asymmetric Cryptography.

[105] 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*, pages 231–244, 2012.

[106] Naofumi Homma, Atsushi Miyamoto, Takafumi Aoki, Akashi Satoh, and Adi Shamir. Collision-based power analysis of modular exponentiation using chosen-message pairs. In *Proceeding Sof the 10th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '08, pages 15–29, Berlin, Heidelberg, 2008. Springer-Verlag.

[107] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293, Oct 2011.

[108] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[109] Vincent Immler, Robert Specht, and Florian Unterstein. Your rails cannot hide from localized EM: how dual-rail logic fails on FPGAs. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 403–424, 2017.

[110] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 308–327, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[111] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 463–481, 2003.

[112] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenada. Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA. In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 129–143. Springer, 2002.

[113] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenada. A practical countermeasure against address-bit differential power analysis. In C. D. Walter, Ç. K. Koç, and C. Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 382–396. Springer, 2003.

[114] Masami Izumi, Jun Ikegami, Kazou Sakiyama, and Kazou Ohta. Improved countermeasures against address-bit DPA for ECC scalar multiplication. In G. De Michell, B. M. Al-Hashimi, W. Müller, and E. Macii, editors, *DATE 2010*, pages 981–984. IEEE, 2010.

[115] Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems — CHES 2017*, volume 10529, pages 623–643, 2017. `https://eprint.iacr.org/2017/637.pdf`.

[116] Marc Joye, Pascal Manet, and Jean-Baptiste Rigaud. Strengthening hardware AES implementations against fault attacks. *IET Information Security*, 1(3):106–110, 2007.

[117] Marc Joye and Sung-Ming Yen. The Montgomery powering ladder. In B. S. Kaliski Jr., editor, *CHES 2002*, volume 2523 of *LNCS*, pages 291–302. Springer, 2002.

[118] Dina Kamel, François-Xavier Standaert, and Denis Flandre. Scaling trends of the AES s-box low power consumption in 130 and 65 nm CMOS technology nodes. In *International Symposium on Circuits and Systems (ISCAS 2009), 24-17 May 2009, Taipei, Taiwan*, pages 1385–1388, 2009.

[119] Emilia Käsper and Peter Schwabe. Faster and timing-attack resistant AES-GCM. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems — CHES 2009*, volume 5747, pages 1–17, 2009. `http://cryptojedi.org/users/peter/#aesbs`.

[120] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[121] Lars Knudsen. Deal - a 128-bit block cipher. 06 1998.

[122] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[123] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1–19, 2019.

[124] 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*, pages 104–113, 1996.

[125] 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*, pages 388–397, 1999.

[126] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[127] Robert Könighofer. A fast and cache-timing resistant implementation of the AES. In Tal Malkin, editor, *Topics in Cryptology — CT-RSA 2008*, volume 4964, pages 187–202, 2008. `https://doi.org/10.1007/978-3-540-79263-5_12`.

[128] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[129] A. Kumar, C. Scarborough, A. Yilmaz, and M. Orshansky. Efficient simulation of em side-channel attack resilience. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 123–130, Nov 2017.

[130] Richard J Larsen and Morris L Marx. *An introduction to mathematical statistics and its applications; 5th ed.* Prentice Hall, Boston, MA, 2012.

[131] Hélène Le Bouder, Ronan Lashermes, Yanis Linge, Gaël Thomas, and Jean-Yves Zie. A multi-round side channel attack on aes using belief propagation. In *International Symposium on Foundations and Practice of Security*, pages 199–213. Springer, 2016.

[132] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. A machine learning approach against a masked AES - reaching the limit of side-channel attacks with a learning model. *J. Cryptographic Engineering*, 5(2):123–139, 2015.

[133] Liran Lerman, Romain Poussier, Olivier Markowitch, and François-Xavier Standaert. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. *Journal of Cryptographic Engineering*, 8(4):301–313, Nov 2018.

[134] Itamar Levi, Davide Bellizia, and François-Xavier Standaert. Reducing a masked implementation's effective security order with setup manipulations and an explanation based on externally-amplified couplings. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):293–317, 2019.

[135] Victor Lomné, Thomas Roche, and Adrian Thillard. On the need of randomness in fault attack countermeasures - application to AES. In *FDTC*, pages 85–94. IEEE Computer Society, 2012.

[136] Pei Luo, Yunsi Fei, Liwei Zhang, and A Adam Ding. Side-channel power analysis of different protection schemes against fault attacks on AES. In *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*, pages 1–6. IEEE.

[137] Jonas Maebe, Ronald De Keulenaer, Bjorn De Sutter, and Koen De Bosschere. Mitigating smart card fault injection with link-time code rewriting: A feasibility study. volume 7859, pages 221–229, 04 2013.

[138] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

[139] Tal G. Malkin, François-Xavier Standaert, and Moti Yung. A comparative cost/security analysis of fault attack countermeasures. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography*, pages 159–172, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[140] Stefan Mangard and Kai Schramm. Pinpointing the side-channel leakage of masked AES hardware implementations. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, pages 76–90, 2006.

[141] Zdenek Martinasek, Jan Hajny, and Lukas Malina. Optimization of power analysis using neural network. In *International Conference on Smart Card Research and Advanced Applications*, pages 94–107. Springer, 2013.

[142] Zdenek Martinasek and Vaclav Zeman. Innovative method of the power analysis. *Radioengineering*, 22(2):586–594, 2013.

[143] Mitsuru Matsui and Atsuhiro Yamagishi. A new method for known plaintext attack of feal cipher. In Rainer A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT' 92*, pages 81–91, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[144] Philippe Maurine. *Securing SoCs in advanced technologies,* `https://cosade.telecom-paristech.fr/presentations/invited2.pdf`.

[145] Dimitrios Mavroeidis, Lejla Batina, Twan van Laarhoven, and Elena Marchiori. Pca, eigenvector localization and clustering for side-channel attacks on cryptographic hardware devices. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 253–268, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[146] David May, Henk L. Muller, and Nigel P. Smart. Random register renaming to foil DPA. In *CHES 2001*, pages 28–38, 2001.

[147] Santos Merino Del Pozo and François-Xavier Standaert. Blind source separation from single measurements using singular spectrum analysis. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, pages 42–59, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[148] Thomas S. Messerges. Securing the aes finalists against power analysis attacks. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, pages 150–164, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[149] Thomas S. Messerges and Ezzy A. Dabbish. Investigations of power analysis attacks on smartcards. In *Proceedings of the 1st Workshop on Smartcard Technology, Smartcard 1999, Chicago, Illinois, USA, May 10-11, 1999*, 1999.

[150] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Power analysis attacks of modular exponentiation in smartcards. In *CHES'99*, pages 144–157, 1999.

[151] N. Moro, K. Heydemann, A. Dehbaoui, B. Robisson, and E. Encrenaz. Experimental evaluation of two software countermeasures against fault attacks. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 112–117, May 2014.

[152] M. Nassar, Y. Souissi, S. Guilley, and J. L. Danger. RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1173–1178, March 2012.

[153] Kashif Nawaz, Dinal Kamel, François-Xavier Standaert, and Denis Flandre. Scaling trends for dual-rail logic styles against side-channel attacks: A case-study. In *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, pages 19–33, 2017.

[154] 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*, pages 529–545, 2006.

[155] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical second-order DPA attacks for masked smart card implementations of block ciphers. In *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, pages 192–207, 2006.

[156] Hoda Pahlevanzadeh, Jaya Dofe, and Qiaoyan Yu. Assessing CPA resistance of AES with different fault tolerance mechanisms. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 661–666. IEEE.

[157] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[158] Maria Panagiotou, Kostas Papagiannopoulos, Jos H. T. Rohling, Johanna H. Meijer, and Tom Deboer. How old is your brain? slow-wave activity in non-rapid-eye-movement sleep as a marker of brain rejuvenation after long-term exercise in mice. *Frontiers in Aging Neuroscience*, 10:233, 2018.

[159] Konstantinos Papagiannopoulos and Aram Verstegen. Speed and size-optimized implementations of the PRESENT cipher for tiny AVR devices. In *Radio Frequency Identification - Security and Privacy Issues 9th International Workshop, RFIDsec 2013, Graz, Austria, July 9-11, 2013, Revised Selected Papers*, pages 161–175, 2013.

[160] Kostas Papagiannopoulos. Low randomness masking and shuffling: An evaluation using mutual information. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):524–546, Aug. 2018.

[161] Kostas Papagiannopoulos and Nikita Veshchikov. Mind the gap: Towards secure 1st-order masking in software. In *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, pages 282–297, 2017.

[162] Kostas Papapagiannopoulos. High throughput in slices: The case of present, PRINCE and KATAN64 ciphers. In *Radio Frequency Identification: Security and Privacy Issues - 10th International Workshop, RFIDSec 2014, Oxford, UK, July 21-23, 2014, Revised Selected Papers*, pages 137–155, 2014.

[163] Sikhar Patranabis, Abhishek Chakraborty, and Debdeep Mukhopadhyay. Fault tolerant infective countermeasure for AES. In *SPACE '15*, pages 190–209, 2015.

[164] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[165] S. Picek, B. Ege, K. Papagiannopoulos, L. Batina, and D. Jakobovic. Optimality and beyond: The case of 4x4 s-boxes. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 80–83, May 2014.

[166] Stjepan Picek, Kostas Papagiannopoulos, Barış Ege, Lejla Batina, and Domagoj Jakobovic. Confused by confusion: Systematic evaluation of DPA resistance of various s-boxes. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology – INDOCRYPT 2014*, pages 374–390, Cham, 2014. Springer International Publishing.

[167] Thomas Popp and Stefan Mangard. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, pages 172–186, 2005.

[168] Axel Poschmann. Lightweight cryptography - cryptographic engineering for a pervasive world. Cryptology ePrint Archive, Report 2009/516, 2009. `http://eprint.iacr.org/`.

[169] Emmanuel Prouff. DPA attacks and s-boxes. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption*, pages 424–441, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[170] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology — EUROCRYPT 2013*, volume 7881, pages 142–159, 2013. `http://www.iacr.org/archive/eurocrypt2013/78810139/78810139.pdf`.

[171] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptology ePrint Archive*, 2018:53, 2018.

[172] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[173] P. Rauzy and S. Guilley. Countermeasures against high-order fault-injection attacks on crt-rsa. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 68–82, Sep. 2014.

[174] Pablo Rauzy, Sylvain Guilley, and Zakaria Najm. Formally proved security of assembly code against power analysis: A case study on balanced logic. *CoRR*, abs/1506.05285, 2015.

[175] Francesco Regazzoni, Luca Breveglieri, Paolo Ienne, and Israel Koren. Interaction between fault attack countermeasures and the resistance against power analysis attacks. In *Fault Analysis in Cryptography. '12*, pages 257–272. 2012.

[176] Francesco Regazzoni, Thomas Eisenbarth, Luca Breveglieri, Paolo Ienne, and Israel Koren. Can knowledge regarding the presence of countermeasures against fault attacks simplify power attacks on cryptographic devices? In *2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, pages 202–210. IEEE.

[177] Francesco Regazzoni, Thomas Eisenbarth, Johann Grobschadl, Luca Breveglieri, Paolo Ienne, Israel Koren, and Christof Paar. Power attacks resistance of cryptographic s-boxes with added error detection circuits. In *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, pages 508–516. IEEE.

[178] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 109–128, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[179] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 109–128, 2011.

[180] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic*

Hardware and Embedded Systems, CHES 2010, pages 413–427, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[181] 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*, pages 171–188, 2009.

[182] Niels Samwel, Lejla Batina, Guido Bertoni, Joan Daemen, and Ruggero Susella. Breaking ed25519 in wolfssl. In *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, pages 1–20, 2018.

[183] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. volume 3659, pages 30–46, 08 2005.

[184] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. *Simple Photonic Emission Analysis of AES*, pages 41–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

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

[186] Tobias Schneider, Amir Moradi, François-Xavier Standaert, and Tim Güneysu. Bridging the gap: Advanced tools for side-channel leakage estimation beyond Gaussian templates and histograms. In Roberto Avanzi and Howard Heys, editors, *Selected Areas in Cryptography — SAC 2016*, volume 10532, pages 58–78, 2017. `https://eprint.iacr.org/2016/719.pdf`.

[187] Peter Schwabe and Ko Stoffelen. All the AES you need on Cortex-M3 and M4. In *Selected Areas in Cryptography — SAC 2016*, 2016. `https://eprint.iacr.org/2016/714.pdf`.

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

[189] Danilo Sijacic, Josep Balasch, Bohan Yang, Santosh Ghosh, and Ingrid Verbauwhede. Towards efficient and automated side channel evaluations at design time. In Lejla Batina, Ulrich K\"uhne, and Nele Mentens, editors, *PROOFS 2018. 7th International Workshop on Security Proofs for Embedded Systems*, volume 7 of *Kalpa Publications in Computing*, pages 16–31. EasyChair, 2018.

[190] Thierry Simon, Lejla Batina, Joan Daemen, Vincent Grosso, Pedro Maat Costa Massolino, Kostas Papagiannopoulos, Francesco Regazzoni, and Niels Samwel.

Towards lightweight cryptographic primitives with built-in fault-detection. Cryptology ePrint Archive, Report 2018/729, 2018. https://eprint.iacr.org/2018/729.

[191] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[192] Etienne Sobole. Cycle counter for Cortex-A8. http://pulsar.webshaker.net/ccc/index.php?lng=us. Retrieved June 2017.

[193] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending sat solvers to cryptographic problems. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 244–257. Springer, 2009.

[194] R. Specht, V. Immler, F. Unterstein, J. Heyszl, and G. Sig. Dividing the threshold: Multi-probe localized em analysis on threshold implementations. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 33–40, April 2018.

[195] Robert Specht, Johann Heyszl, Martin Kleinsteuber, and Georg Sigl. Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution EM measurements. In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, pages 3–19, 2015.

[196] Robert Specht, Johann Heyszl, and Georg Sigl. Investigating measurement methods for high-resolution electromagnetic field side-channel analysis. In *2014 International Symposium on Integrated Circuits (ISIC), Singapore, December 10-12, 2014*, pages 21–24, 2014.

[197] 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*, pages 443–461, 2009.

[198] François-Xavier Standaert, Eric Peeters, Cédric Archambeau, and Jean-Jacques Quisquater. Towards security limits in side-channel attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, pages 30–45, 2006.

[199] 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*, pages 99–134. Springer, 2010.

[200] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The

world is not enough: Another look on second-order DPA. In Masayuki Abe, editor, *Advances in Cryptology — ASIACRYPT 2010*, volume 6477, pages 112–129, 2010. http://www.iacr.org/archive/asiacrypt2010/6477112/6477112.pdf.

[201] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order dpa. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, pages 112–129, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[202] Ko Stoffelen. Optimizing s-box implementations for several criteria using SAT solvers. *IACR Cryptology ePrint Archive*, 2016:198, 2016.

[203] Marc Stöttinger. *Mutating runtime architectures as a countermeasure against power analysis attacks*. PhD thesis, Darmstadt University of Technology, Germany, 2012.

[204] Takeshi Sugawara, Daisuke Suzuki, Minoru Saeki, Mitsuru Shiozaki, and Takeshi Fujino. On measurable side-channel leaks inside ASIC design primitives. *J. Cryptographic Engineering*, 4(1):59–73, 2014.

[205] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[206] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[207] Keccak team. Noekeon cipher. http://keccak.noekeon.org.

[208] Keccak team. Note on side-channel attacks and their countermeasures. http://keccak.noekeon.org/NoteSideChannelAttacks.pdf.

[209] Mike Tunstall, Louiza Papachristodoulou, and Kostas Papagiannopoulos. Boolean exponent splitting, under submission.

[210] Harshal Tupsamudre, Shikha Bisht, and Debdeep Mukhopadhyay. Destroying fault invariant with randomization - A countermeasure for AES against differential fault attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, pages 93–111, 2014.

[211] 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. *COSADE*, 2017:272, 2017.

[212] Florian Unterstein, Johann Heyszl, Fabrizio De Santis, Robert Specht, and Georg Sigl. High-resolution EM attacks against leakage-resilient PRFs explained - and an improved construction. Cryptology ePrint Archive, Report 2018/055, 2018. https://eprint.iacr.org/2018/055.

[213] Harry L. Van Trees. *Detection, Estimation, and Modulation Theory: Part IV: Optimum Array Processing*. John Wiley and Sons, Inc., 2002.

[214] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In Aggelos Kiayias, editor, *Topics in Cryptology — CT-RSA 2011*, pages 104–119, 2011. https://doi.org/10.1007/978-3-642-19074-2_8.

[215] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In *Advances in Cryptology — ASIACRYPT 2014*, volume 8873, pages 282–296, 2014. https://eprint.iacr.org/2014/410.pdf.

[216] 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*, pages 740–757, 2012.

[217] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Mutual information analysis: How, when and why? In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 429–443, 2009.

[218] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[219] Junwei Wang, Praveen Kumar Vadnala, Johann Großschädl, and Qiuliang Xu. Higher-order masking in practice: A vector implementation of masked AES for ARM NEON. In Kaisa Nyberg, editor, *Topics in Cryptology — CT-RSA 2015*, volume 9048, pages 181–198, 2015. http://dx.doi.org/10.1007/978-3-319-16715-2_10.

[220] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, USA, 4th edition, 2010.

[221] Carolyn Whitnall, Elisabeth Oswald, and Luke Mather. An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis. In *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, pages 234–251, 2011.

[222] Shuguo Yang, Yongbin Zhou, Jiye Liu, and Danyang Chen. Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations. In *International Conference on Information Security and Cryptology*, pages 169–185. Springer, 2011.

[223] Liwei Zhang, A. Adam Ding, Francois Durvaux, Francois-Xavier Standaert, and Yunsi Fei. Towards sound and optimal leakage detection procedure (extended version). In *Smart Card Research and Advanced Applications — CARDIS 2017*, 2017. `https://eprint.iacr.org/2017/287`.

[224] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *SCIENCE CHINA Information Sciences*, 58(12):1–15, 2015.

[225] Shoshana Zuboff. *The age of surveillance capitalism: the fight for the future at the new frontier of power*. 2018.