ICTEAM Institute
UCL Crypto Group

NXP Semiconductors

Shortcut Side-Channel Security Evaluations: Application to Elliptic Curve Cryptography, Masking and Shuffling.

Melissa Azouaoui

Thèse présentée en vue de l'obtention du grade de
docteur en sciences de l'ingénieur.

Composition du jury :
Pr. Laurent Francis (UCL) - Président du jury
Pr. François-Xavier Standaert (UCL) - Promoteur
Pr. Lejla Batina (RU)
Dr. François Koeune (UCL)
Dr. Romain Poussier (NTU)
Dr. Matthieu Rivain (CryptoExperts)
Dr. Vincent Verneuil (NXP)

Hambourg, Allemagne – 2021

Acknowledgments

First and foremost, I am deeply grateful to my PhD supervisors: Pr. François-Xavier Standaert and Dr. Vincent Verneuil for granting me this opportunity and the honor to work alongside them. Thanks to their guidance I have grown so much as a researcher and as a person as well. I would also like to extend a special thank you to Dr. Romain Poussier for his help at the beginning of this thesis with papers, explanations, discussions, datasets and tools.

I would also like to express my gratitude to members of the jury committee. Pr. Laurent Francis for presiding it, Pr. Lejla Batina, Dr. François Koeune and Dr. Matthieu Rivain for their special interest, discussions, knowledge and recommendations.

I am very fortunate to have been and continue to be surrounded and collaborating with many wonderful, humble and kind people. First, I would like to offer my sincere thanks to my friend and co-author Kostas Papagiannopoulos for the many discussions, pleasant and fun times. I am also very grateful to my co-authors: François Durvaux, Vincent Grosso, Martin Butkus and Dominik Zürner, and in particular Olivier Bronchain for always answering my questions and the valuable discussions.

I would also like to acknowledge the memorable time spent with all the bright and kind-hearted people who participated in the REASSURE project that I did not name already: Valentina Banciu, Davide Bellizia, Ileana Buhan, Lukasz Chmielewski, Nicolas Debande, Sébastien Duval, Pr. Elisabeth Oswald, Adrian Thillard and Carolyn Whitnall.

I would like to thank Stefan and Thomas for welcoming me in their respective teams at my start at NXP, and thank you to Julia and Denise for their support and help. A special thanks goes to Joppe, along with my new teammates, in particular Tobias, Christine, Joost and Babette, for welcoming me into a new team and offering me the opportunity to

work on new and exciting research challenges.

A special gratitude goes to Pr. Louis Goubin, Dr. Michaël Quisquater, Dr. Luca De Feo and Dr. Christina Boura for introducing me to cryptography and side-channel attacks. I would also like to thank Dr. Leila Kloul and Dr. Theirry Mautor for their kind help. Thank you to Gilles Macario-Rat for his kindness and for giving me the first opportunity to work on side-channel attacks.

Naturally, I would like to thank my family, my dad and mom, for their support and encouragements, and my sister and brother for simply being their true best selves. Last but not least, I would like to thank my other half and best friend Leif, for always being there for me, making me happy, smile and laugh.

Abstract

Since 1996, numerous attacks have been shown to uncover secrets by exploiting a device's physical behavior or emanation, such as power consumption, electromagnetic radiation, execution time, or temperature. Until then, the adversary was always assumed to be constrained by black box assumptions, but we are aware now that he is considerably more powerful than expected. These new attacks, called side-channel attacks, have led to the design of several countermeasures to protect cryptographic secrets. This raises the need for sound methods to assess their security. However, this is a challenging task and the difficulty resides in the fact that unsuccessful attacks do not prove security. We must expand our scope to more powerful adversaries to reach higher security guarantees. In this thesis, we investigate such worst-case evaluations in the context of both symmetric and asymmetric cryptography. In the first part of the thesis, we speed up evaluations of elliptic curve cryptography implementations against horizontal attacks using shortcut formulas. In the second part, we study worst-case analytical belief-propagation-based attacks and compare them to simpler divide-and-conquer attacks to evaluate the elliptic curve point randomization countermeasure. The third part of the thesis relates to the masking and shuffling countermeasures. In this respect, we first improve tools for their evaluations, then we analyze and enhance the security impact of their combination.

Contents

List of acronyms

AES	Advanced Encryption Standard
BP	Belief Propagation
CC	Common Criteria
CDF	Cumulative Distribution Function
CPA	Correlation Power Analysis
CRT	Chinese Remainder Theorem
DFA	Differential Fault Attack
DHP	Diffie-Hellman Problem
DLP	Discrete Logarithm Problem
DPA	Differential Power Analysis
D&C	Divide-and-Conquer
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECSM	Elliptic Curve Scalar Multiplication
E&P	Extend-and-Prune
FI	Fault Injection
FIPS	Federal Information Processing Standard
GE	Guessing Entropy
HD	Hamming Distance
HDPA	Horizontal Differential Power Analysis
HO	High-Order
HW	Hamming Weight

IFA	Ineffective Fault Attack
IOL	Independence of Operations Leakages
ISW	Ishai Sahai Wagner
IT	Information Theoretic
LDA	Linear Discriminant Analysis
LIM	Long Integer Multiplication
LRA	Linear Regression Attack
LRPM	Local Random Probing Model
MI	Mutual information
MIA	Mutual Information Analysis
NIST	National Institute of Standards and Technology
OTA	Online Template Attack
PCA	Principal Component Analysis
PDF	Probability Density Function
PI	Perceived Information
POI	Point Of Interest
PQC	Post-Quantum Cryptography
RNS	Redundant Number System
RPM	Random Probing Model
RSA	Rivest–Shamir–Adleman
SASCA	Soft Analytical Side-channel Attacks
SCA	Side-Channel Attack
SFA	Statistical Fault Attack
SIFA	Statistical Ineffective Fault Attack
SNR	Signal-to-Noise Ratio
SPA	Simple Power Analysis
SPN	Substitution Permutation Network
SR	Success Rate
TVLA	Test Vector Leakage Assessment

List of notations

Algebra

$(x_i)_{0 \leq i < n}$	set of n elements/values
$\lvert x \rvert$	size of x in bits
$\mathbf{v}$	vector (bold lowercase)
$\mathbf{v}[i]$	i^{th} element of vector $\mathbf{v}$
$\mathbf{M}$	matrix (bold uppercase)
$\mathbf{M}^{\mathrm{T}}$	matrix transpose
$\mathbb{F}_q$	finite field with q elements
x^{-1}	multiplicative inverse of x
$\oplus$	XOR or bit-wise addition over $\mathbb{F}_{2^n}$
$\otimes$	multiplication or bit-wise AND over $\mathbb{F}_{2^n}$
$\mathrm{ord}(G)$	order of element G
$\langle G \rangle$	cyclic subgroup generated by G
$\mathcal{E}(\mathbb{F}_q)$	set of points on an elliptic curve over $\mathbb{F}_q$
f	function or metric (sans serif font)
$\sim$	equivalence relation
$\%$	modulus operator
$/$	integer division

Statistics

X	random variable (uppercase letter)
x	realization of X (lowercase letter)
$x \xleftarrow{\$} \mathcal{X}$	pick x randomly and uniformly from set $\mathcal{X}$
$x \sim \mathcal{D}$	x is distributed according to distribution $\mathcal{D}$
$\Pr[A]$	probability of event A
$\Pr[A\|B]$	conditional probability of A given B
$\mathbb{E}$	expected value or mean operator
$\mathbb{V}$	variance operator
$\mathbb{C}ov$	covariance operator
$\mathcal{U}(\mathbb{F})$	uniform distribution over $\mathbb{F}$
$\mathcal{N}(\mu, \sigma^2)$	gaussian distribution with mean μ and variance σ^2
Φ	CDF of the standard normal distribution
ρ	Pearson's correlation coefficient
$\mathrm{H}(X)$	entropy of random variable X

List of publications

1. Melissa Azouaoui, Olivier Bronchain, Vincent Grosso, Kostas Papagiannopoulos, François-Xavier Standaert. *Bitslice Masking and Improved Shuffling: How and When to Mix Them in Software?*

2. Melissa Azouaoui, Kostas Papagiannopoulos, Dominik Zürner. *Blind Side-Channel SIFA.* - DATE 2021

3. Melissa Azouaoui, François Durvaux, Romain Poussier, François-Xavier Standaert, Kostas Papagiannopoulos, Vincent Verneuil. *On the Worst-Case Side-Channel Security of ECC Point Randomization in Embedded Devices.* - INDOCRYPT 2020

4. Melissa Azouaoui, Davide Bellizia, Ileana Buhan, Nicolas Debande, Sébastien Duval, Christophe Giraud, Éliane Jaulmes, François Koeune, Elisabeth Oswald, François-Xavier Standaert, Carolyn Whitnall. *A Systematic Appraisal of Side Channel Evaluation Strategies.* - SSR 2020

5. Melissa Azouaoui, Romain Poussier, François-Xavier Standaert, Vincent Verneuil. *Key Enumeration from the Adversarial Viewpoint: When to Stop Measuring and Start Enumerating?* - CARDIS 2019

6. Melissa Azouaoui, Romain Poussier, François-Xavier Standaert. *Fast Side-Channel Security Evaluation of ECC Implementations - Shortcut Formulas for Horizontal Side-Channel Attacks Against ECSM with the Montgomery Ladder.* - COSADE 2019

7. Melissa Azouaoui, Vincent Verneuil. *Attaques par canaux auxiliaires sur AES – Partie 2.* - MISC n°98 2018

8. Melissa Azouaoui, Vincent Verneuil. *Attaques par canaux auxiliaires sur AES – Partie 1.* - MISC n°97 2018

Chapter 1

Introduction

In 1996, Paul Kocher introduced a new kind of attack on cryptographic algorithms called Side-Channel Attacks (SCA). He showed that physical implementations of cryptographic algorithms (e.g., the Rivest–Shamir–Adleman (RSA) cryptosystem) are vulnerable to attacks exploiting observations of the algorithm's execution, such as the running time [99]. Following in 1998, Kocher, Jaffe and Jun showed this time that statistical analysis of physical observations such as power consumption can be exploited to find the secret key of most implementations of private- and public-key cryptographic algorithms [100]. As a result, it has become clear that we should pay careful attention to the implementation of cryptographic algorithms on devices which an attacker can have physical access to, such as embedded devices. Numerous countermeasures have been proposed, e.g., random delay insertion [44], shuffling [83] and masking [88] for symmetric cryptography, and many randomization countermeasures [43] for asymmetric cryptography. Research on both attacks and countermeasures is an ongoing effort shared by both the academic and industrial communities.

1.1 Context and motivation

Detecting and exploiting side-channel vulnerabilities is a critical part of security evaluations of embedded cryptography. Within the REASSURE consortium [3], we have analyzed different side-channel evaluation methodologies and discussed the optimality of the steps of a side-channel attack [7]. In particular, we considered common evaluation regimes which can be categorized into *Detect-&-then-stop* approaches, such as the Federal Information Processing Standards (FIPS) 140 standard [129], and *Detect-&-then-attack* approaches such as the Common

Criteria (CC) framework [4]. On one hand, the FIPS standard aims for cheaper evaluations by following a conformance style testing by using leakage detection tests, e.g., the popular Test Vector Leakage Assessment (TVLA) [69]. However, the main drawback of these evaluations is that leakage detection which is based on statistical hypothesis testing can only show the presence of a leak but cannot certify security [153, 169]. On the other hand, the CC framework is attack-driven and is based on a list of attacks to consider during an evaluation and a rating system based on the difficulty of applying these attacks to the target under evaluation. The choice of attacks to mount during the evaluation period is left to the evaluator's judgment.

Both the optimality and the soundness of side-channel exploitation techniques are important aspects, since they impact the confidence an evaluator has on the outcome of a security evaluation. The REASSURE discussion in [7], and research in the field, e.g., the work of Bronchain and Standaert [28], suggest that security confidence can only be achieved when considering worst-case adversaries. Accordingly, a possible rational evaluation strategy is to approach as closely as possible the worst-case adversary. Based on this observation, the REASSURE consortium has suggested a new *backwards evaluation* approach [7].

Worst-case adversary. The backwards evaluation hinges on the description of a worst-case adversary, which is defined as having the following capabilities or knowledge. First, during the profiling phase of the attack, full control over the algorithm's inputs, e.g., plaintext, secret key material, and randomness. Second, detailed knowledge of the implementation, e.g., software source code. Additionally, a worst-case adversary exploits multiple leaking time samples and uses multivariate leakage modeling. Moreover, a worst-case adversary aims for an optimal information processing strategy, enhanced with enumeration by leveraging computational power.

This thesis investigates such strategies. Information processing strategies are categorized in two classes: simple strategies and advanced strategies. Simple strategies include Divide-and-Conquer (D&C) attacks in the context of symmetric cryptography and Extend-and-Prune (E&P) attacks in the context of asymmetric cryptography. Advanced strategies include algebraic or analytical attacks such as Soft Analytical Side-Channel Attacks (SASCA) [163].

Backwards evaluation. This approach advocates that the starting point of any evaluation is the specification of a worst-case adversary. While very powerful and nearly unbounded, the definition of a worst-

case adversary is not difficult. This definition only depends on the target device and implementation, and offers a stable starting point for any evaluator. Then, for the evaluation to remain realistic (in terms of the capabilities the device allows) and practical (in terms of time and effort), the adversarial capabilities are relaxed. The consequences of this relaxation are discussed, in particular, with respect to its impact on the attack complexity. Once a practically feasible attack is derived from the previous definition and relaxation, it is demonstrated on the target.

The backwards evaluation approach offers an improvement over common evaluation strategies such as CC. First, it establishes the best attack that can be realistically applied to a target, since its feasibility relies on the attacker's capabilities. Moreover, it offers a considerable speed up of security evaluations, since finding the worst-case attack path is far less challenging than finding a possible attack without exploiting any knowledge of the target. However, while defining the worst-case adversary is simple, defining the best tools (e.g., statistical or modeling tools) to use for each step of the attack remains a complex problem. Some tools might introduce a risk of overstated security, since it is hard to argue about their optimality. In particular, regarding the information processing step, which we mainly deal with in this thesis, while simple D&C and E&P strategies are clear and the ad hoc tools by now, more advanced analytical strategies are a source of sub-optimality. The reason behind this is the fact that these new techniques are not yet well understood and are occasionally based on heuristics, despite being theoretically superior to classic strategies.

Since we are motivated by the challenge of raising confidence in both side-channel evaluations and countermeasures, the main goal of this thesis is to improve worst-case attack strategies. Particularly, and as previously mentioned, we are interested in analyzing and improving different aspects of the information processing step. Moreover, we consider both symmetric and asymmetric cryptography, due to lower interest in the literature in the systematic security analysis of public key cryptography implementations. Additionally, in the field of side-channel analysis, there is a slow translation between research and practice. One goal of this thesis, bringing together UCL and NXP, is to overcome this barrier and to produce knowledge that is not only applicable in the academic field, but would benefit embedded systems designers and manufacturers.

1.2 Contributions and thesis organization

The main contributions in this thesis are threefold.

First, in Chapter 3 we consider worst-case horizontal attacks on Elliptic Curve Cryptography (ECC) implementations [134]. Horizontal attacks exploit multiple leakage points, hence they require a significant modeling/profiling effort. We are motivated by this matter and inspired by the literature on shortcut formulas for the success rate estimation of side-channel attacks against symmetric primitive implementations [139, 156, 109]. In this respect, we present shortcut formulas to estimate the success rate of horizontal attacks against the central ECC operation: the Elliptic Curve Scalar Multiplication (ECSM). These shortcut formulas require minimal profiling effort and are based on practically falsifiable assumptions.

In Chapter 4, we analyze the worst-case security of the point randomization countermeasure. This technique proposed by Coron [43] is an important countermeasure to protect ECC implementations against side-channel attacks. In this respect, we explore both D&C and analytical information processing strategies [163]. Our research shows that due to the ephemeral nature of the point randomization secret, the gain of analytical techniques over D&C attacks is limited. Accordingly, for reasonable noise levels, efficient D&C attacks are sufficient to approach the worst-case security of the point randomization. We then evaluate the noise levels required to protect point randomization and thus hinder horizontal adversaries taking advantage of the input point knowledge to attack the following ECSM operation. We also show that reasonable security can be reached in software by considering simple optimizations that reduce the amount of leakage. Additionally, we compare different elliptic curve coordinate systems with respect to their resistance. Finally, we provide security bounds for targets that are harder to analyze, such as 32-bit implementations without performing time-consuming attacks, by leveraging the efficiency of the Local Random Probing Model (LRPM), introduced by Guo et al. [78].

In Chapter 5, we focus on the two most common countermeasures to protect implementations of the Advanced Encryption Standard (AES), namely masking [33, 71, 120, 88] and shuffling [83]. First of all, we revisit and improve the attacks against shuffling proposed by Veyrat-Charvillon et al. [164]. We then compare D&C attacks against shuffling to worst-case multivariate attacks and reach the conclusion that when the noise level increases the gain between the more simple and more complex attacks decreases. In practice, this results in a speed up of side-channel evaluations of the shuffling countermeasure since no enumeration, analytical or multivariate analysis are required. This is in line with previous conclusions from Chapter 4 of this thesis. Similarly to

the secret parameter of the point randomization in ECC, the permutation used for shuffling is also an ephemeral secret and D&C attacks are generally sufficient to exploit its limited leakage. Moreover, we study the combination of masking and shuffling and identify a new configuration amplifying the impact of shuffling exponentially in the number of masking shares. We then compare this method to an existing and similar work from Rivain, Prouff and Doget [140]. Finally, we discuss the challenges of implementing algorithmic countermeasures such as masking and shuffling under low noise conditions and evaluate how to best combine and prioritize bitslice masking [21, 73, 72] and shuffling to maximize the security-to-cost ratio.

We present at the end of this manuscript two additional contributions relating to key-agnostic rank estimation in the adversarial setting [10] and combined blind side-channel and Statistical Ineffective Fault Attacks (SIFA) [9].

Chapter 2

Background

In this chapter, we provide common background to the following chapters. First, in Section 2.1 we give some background on symmetric cryptography and particularly on the AES. Then, in Section 2.2 we describe some preliminaries to elliptic-curve-based asymmetric cryptography. Next, in Section 2.3 we give an introduction to side-channel attacks, partly based on publications 7 and 8, and some examples of side-channel attacks targeting elliptic curve implementations. Following, in Section 2.4 we describe common countermeasures to protect block ciphers and elliptic curve cryptography implementations against side-channel attacks. Finally, in Section 2.5 we present soft analytical side-channel attacks.

2.1 Advanced encryption standard

Block ciphers. A block cipher is a pair of functions: an encryption function Enc and a decryption function Dec. Both functions are parametrized by the same secret key k and operate on messages of fixed length n. For any plaintext x in $\mathbb{F}_{2^n}$ these two functions verify that $\mathsf{Dec}_k(\mathsf{Enc}_k(x)) = x$. The security of block ciphers is based on the notions of *confusion* and *diffusion* defined by Shannon [150]. Confusion means that the relation between the plaintext and the ciphertext is as complex as possible. Diffusion entails that every bit of the ciphertext depends on many bits of the plaintext and the secret key. Secure block ciphers are designed using repeated transformations that allow for good enough degrees of confusion and diffusion. A relevant family of block ciphers are Substitution Permutation Networks (SPN).

Substitution permutation network. SPNs are algorithms based on the repeated application of a round function to their state, which is initialized to the plaintext. These round functions are composed of substitutions and permutations. Additionally, in each round, a round key is introduced by addition to the algorithm's state. The round keys are derived using a key scheduling algorithm given the master key. The substitution operations and the round key additions provide confusion, whereas the permutations provide diffusion.

Advanced encryption standard. The block cipher Rijndael designed by Daemen and Rijmen [45], was chosen as the Advanced Encryption Standard (AES) by the National Institute of Standards and Technology (NIST) in 2001, and belongs to the family of SPN ciphers. It operates on data blocks of 128 bits split into 16 bytes, but with three possible key lengths: 128, 192 and 256 bits. Based on the key length, the round function is repeated 10, 12 or 14 times, respectively. The structure of an AES round is shown on Figure 2.1. The AES state is represented by a 4×4 matrix where the first column corresponds to the first 4 bytes. Following the key scheduling, to derive the round keys from the master key, the first step is an initial key addition to the plaintext. Then the four AES transformations, namely SubBytes, ShiftRows, MixColumns and AddRoundKey, are applied in order and repeated for the corresponding number of rounds. The final round does not include the MixColumns operation. We describe below the operations involved.

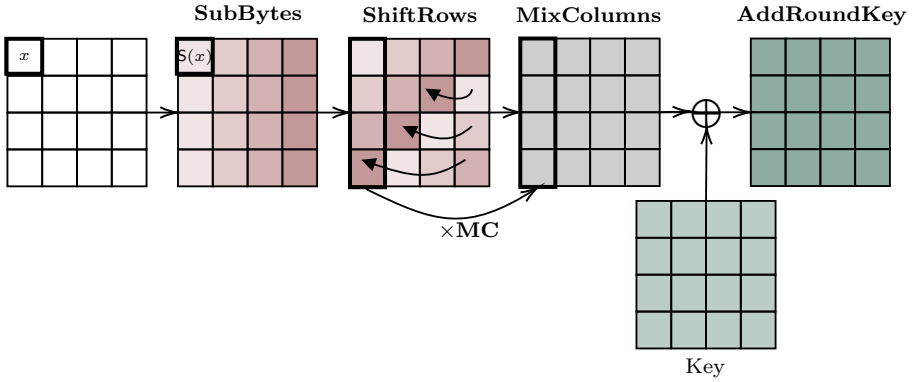


Figure 2.1: AES round function.

– **SubBytes.** The SubBytes is a substitution operation applied to the bytes of the state. It is usually represented by a table or function S. This operation is the non-linear part of the cipher,

since it is the combination of the multiplicative inverse in $\mathbb{F}_{2^8}$ and an affine transformation.

- **ShiftRows.** The ShiftRows is a permutation operation. As illustrated in Figure 2.1, it leaves the first row of the state matrix unaltered and rotates the second row by 1, the third row by 2 and the last row by 3. This is the only operation that diffuses the information among the four columns of the state matrix.

- **MixColumns.** The MixColumns transformation consists in multiplying each column of the state by a fixed matrix. Precisely, each column is represented by a four coefficient polynomial in $\mathbb{F}_{2^8}$ and then multiplied by the polynomial $3x^3 + x^2 + x + 2$ modulo $x^4 + 1$.

- **AddRoundKey.** The AddRoundKey is the key addition operation that introduces the round key into the state. The state and the round key are simply byte-wise XORed.

2.2 Elliptic curve cryptography

2.2.1 Elliptic curve

Let $\mathbb{F}_p$ be a finite field of characteristic $p \neq \{2, 3\}$. An elliptic curve $\mathcal{E}(\mathbb{F}_p)$ over $\mathbb{F}_p$ defines a set of points $(x, y) \in \mathbb{F}_p^2$ which satisfy the short Weierstraß equation:

$$y^2 = x^3 + ax + b$$

with $a, b \in \mathbb{F}_p$ and $4a^3 + 27b^2 \neq 0$, along with the point at infinity O. If $\mathbb{F}_p$ is of characteristic 2, the points on $\mathcal{E}(\mathbb{F}_p)$ satisfy an equation of the type:

$$y^2 + cy = x^3 + ax + b \;\; or \;\; y^2 + xy = x^3 + ax^2 + b$$

The set of points of an elliptic curve, together with the point at infinity O being the neutral element, equipped with an addition operation form an abelian group. The addition of two distinct points is denoted by $+$ and the k-times repeated addition $P + P + ... + P$ by $k \cdot P$, called the scalar multiplication. When the same point P is added to itself, then this operation is referred to as point doubling and the result is denoted by $2P$. The opposite of a point $P = (x, y)$ is $-P = (x, -y)$, and verifies that $P + (-P) = O$. In this thesis we restrict ourselves to curves over fields of prime characteristic $p > 3$.

The number of points on an elliptic curve is called the order and is denoted by $\text{ord}(\mathcal{E}(\mathbb{F}_p))$. It can be computed in polynomial time using

Schoof's algorithm [148], which combines the Chinese Remainder Theorem (CRT) and Hasse's theorem [81] which bounds the order of an elliptic curve and states that:

Hasse's theorem. If $\mathcal{E}(\mathbb{F}_p)$ is an elliptic curve over a finite field with p elements then:

$$|p + 1 - \text{ord}(\mathcal{E}(\mathbb{F}_p))| \leq 2\sqrt{p}$$

For any point $P \in \mathcal{E}(\mathbb{F}_p)$, the set of its multiples forms a cyclic subgroup of $\mathcal{E}(\mathbb{F}_p)$ denoted by $\langle P \rangle$. The order of this subgroup and equivalently the order of P is denoted as $\text{ord}(P)$ and is defined as the smallest integer n such that $n.P = O$. Such a cyclic group of large order can be used for cryptographic applications, and a given generator of this group is often called a *base point*.

2.2.2 Elliptic curve cryptography systems

Elliptic curve cryptography was introduced by both Koblitz [98] and Miller [122] independently and is based on the intractability of the Discrete Logarithm Problem (DLP) in the group of elliptic curve points, which is defined as:

DLP. Let $\mathcal{G}$ be a finite cyclic group generated by G, i.e. $\mathcal{G} = \langle G \rangle$. Given G and Q in $\mathcal{G}$, find x in $\{0, 1, \ldots, \text{ord}(G) - 1\}$ such that $Q = x \cdot G$. The value of x is the discrete logarithm of Q in base G.

2.2.2.1 Elliptic Curve Diffie-Hellman (ECDH)

One of the main applications of elliptic curve cryptography is the Diffie-Hellman key exchange. It is shown on Figure 2.2, where Alice and Bob agree on a shared secret key k. On the diagram, red variables are secret and green variables are public or can be easily observed, e.g., the public base point G. The secret key k can later be used for symmetric encryption between Alice and Bob. This secret key establishment cannot be compromised by an adversary only observing (and unable to alter) the exchange between Alice and Bob, unless additionally able to solve the Diffie-Hellman Problem (DHP) which is defined as:

DHP. Let $\mathcal{G}$ be a finite cyclic group. Given G, $a \cdot G$ and $b \cdot G$ in $\mathcal{G}$ with a, b in $\{0, 1, \ldots, \text{ord}(G) - 1\}$, compute $a \cdot b \cdot G$.

2.2.2.2 Elliptic Curve Digital Signature Algorithm (ECSDA)

Another application of elliptic curve cryptography are digital signatures. To start signing messages, the first step for Alice is to generate her secret

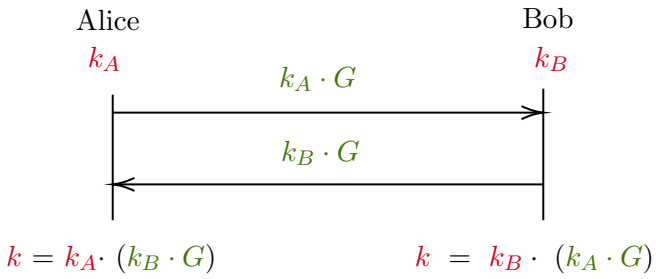


Figure 2.2: The Elliptic Curve Diffie-Hellman (ECDH) key exchange.

key $a \xleftarrow{\$} \{1, 2, \ldots, \mathrm{ord}(G)\}$ and her public key $Q = a \cdot G$. Then, she can compute the signature (r, s) of a message m following Algorithm 1 and transmit both the message and the signature to Bob. The signature algorithm uses a hash function H and for each new signature a fresh random secret nonce k, also called *ephemeral key*. Once the message and corresponding signature are received by Bob, he can verify that the message was indeed signed by Alice using Algorithm 2. Valid signatures pass the verification check since $s^{-1}(\mathsf{H}(m) \cdot G + r \cdot Q) = k \cdot (\mathsf{H}(m) + a \cdot x)^{-1} \cdot (\mathsf{H}(m) + a \cdot x) \cdot G = k \cdot G = (x, y)$.

Algorithm 1 Signature

Input secret key a, message m
Output signature (r, s)

1: $k \xleftarrow{\$} \{1, 2, \ldots, \mathrm{ord}(G) - 1\}$
2: $(r, y) \leftarrow k \cdot G$
3: **if** $r \ \% \ \mathrm{ord}(G) = 0$ **then**
4: **go to** line 1
5: $s \leftarrow k^{-1}(\mathsf{H}(m) + a.r) \ \% \ \mathrm{ord}(G)$
6: **if** $s \ \% \ \mathrm{ord}(G) = 0$ **then**
7: **go to** line 1
8: **return** (r, s)

2.2.3 Elliptic curve scalar multiplication

The main operation in elliptic curve cryptography systems is the Elliptic Curve Scalar Multiplication (ECSM) that computes efficiently $k \cdot P$, $\forall k \in \mathbb{F}_p^*$ and $\forall P \in \mathcal{E}(\mathbb{F}_p)$. The variable k is usually a secret, e.g., the secret key in ECDH or a secret ephemeral nonce in ECDSA. In the following, let $k = (k_0, ..., k_{l-1})$ the binary representation of the scalar k, where k_0 is the most significant bit and $l = |k|$ is the length of k in

Algorithm 2 Verification

Input message m, signature (r, s)
Output **valid** or **invalid** signature

1: **if** $r, s \notin \{1, 2, \ldots, \text{ord}(G) - 1\}$ **then**
2: **return invalid**
3: $(x, y) \leftarrow s^{-1} \cdot (\mathsf{H}(m) \cdot G + r \cdot Q)$
4: **if** $((x, y) = O)$ or $(x \ \% \ \text{ord}(G) \neq r)$ **then**
5: **return invalid**
6: **return valid**

bits. The scalar multiplication can be performed in different ways. In the following, we provide a few examples that are the most relevant for the following chapters.

2.2.3.1 Double-and-Add

First, the straightforward method to perform a scalar multiplication is the Double-and-Add algorithm. It is analogous to the Square-and-Multiply algorithm to perform an exponentiation in a multiplicative group used for RSA. It is described in Algorithm 3.

Algorithm 3 Double-and-add

Input P , $k = (k_0, ..., k_{l-1})$
Output $k \cdot P$

1: $Q \leftarrow O$
2: **for** $i = 0$ to $l - 1$ **do**
3: $Q \leftarrow 2Q$
4: **if** $k_i = 1$ **then**
5: $Q \leftarrow Q + P$
6: **return** Q

The Double-and-Add algorithm always performs an addition when the current scalar bit is equal to one. This algorithm is hence not time constant and leaks the HW of the secret scalar through timing side channel. Additionally, it leaks the values of the scalar bits through other kinds of side channels, e.g., power or electromagnetic side channels.

2.2.3.2 Double-and-Add-Always

Due to the weakness of the Double-and-Add algorithm in front of simple side-channel attacks, Coron recommended the use of the Double-and-Add-Always algorithm [43]. This scalar multiplication algorithm always

performs the same sequence of operations no matter the bit values of k. It is described in Algorithm 4, where T is a dummy variable that is discarded at the end of the computation.

Algorithm 4 Double-and-Add-Always

Input P , $k = (k_0, ..., k_{l-1})$
Output $k \cdot P$

1: $Q \leftarrow O$
2: **for** $i = 0$ to $l - 1$ **do**
3: $\quad Q \leftarrow 2Q$
4: $\quad$ **if** $k_i = 1$ **then**
5: $\quad\quad Q \leftarrow Q + P$
6: $\quad$ **else**
7: $\quad\quad T \leftarrow Q + P$
8: **return** Q

2.2.3.3 Montgomery ladder

Another algorithm that always performs the same sequence of operations is the Montgomery ladder [124, 93], described in Algorithm 5. As opposed to the Double-and-Add-Always algorithm, the Montgomery ladder does not require any dummy variables or dummy operations. Additionally, while its complexity is theoretically the same as for the Double-and-Add-Always algorithm, it was shown to speed up the scalar multiplication [124, 26, 89, 60]. In particular, Fischer et al. [60] present a fast parallel scalar multiplication when the y-coordinate of the result is not needed. Additionally, the invariant $Q_0 - Q_1 = P$ at every iteration in the Montgomery ladder offers a convenient check for faults.

Algorithm 5 Montgomery ladder

Input P , $k = (k_0, ..., k_{l-1})$
Output kP

1: $Q_0 \leftarrow O$
2: $Q_1 \leftarrow P$
3: **for** $i = 0$ to $l - 1$ **do**
4: $\quad Q_{1-k_i} \leftarrow Q_0 + Q_1$
5: $\quad Q_{k_i} \leftarrow 2Q_{k_i}$
6: **return** Q_0

We provided a few examples of scalar multiplication algorithms that process the scalar bits from most significant to least significant. Most

scalar multiplication algorithms can be modified to process the bits from least to most significant. Additionally, on top of the examples described above, many other algorithms can be used to implement scalar multiplication. Mainly, m-ary and windowed methods allow to process multiple bits of the scalar at the same time [70]. We do not detail these methods since they are not relevant for the remaining chapters in this manuscript. For a survey and detailed comparison of scalar multiplication algorithms we refer to [160].

2.2.4 Coordinate systems and point arithmetic

A scalar multiplication involves operations on the points of the curve, additions and doublings in $\mathcal{E}(\mathbb{F}_p)$, which in turn involve operations on the coordinates of the points in $\mathbb{F}_p$. Point operations depend on the kind of coordinate system used to represent the points. In the following we detail the most common of these representations, namely affine, homogeneous and Jacobian coordinate systems, and detail the corresponding point operations.

2.2.4.1 Affine coordinates

Let $P_1, P_2 \in \mathcal{E}(\mathbb{F}_p)$ two points on an elliptic curve with $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, such that $y_1 \neq 0$ and $P_1 \neq \pm P_2$.

- **Point opposite.** The opposite of a point P_1 is $-P_1 = (x_1, -y_1)$.
- **Point addition.** The sum of two points P_1 and P_2 is a point $P_3 = P_1 + P_2 = (x_3, y_3)$ such that:

$$\begin{cases} x_3 = m^2 - x_1 - x_2 \\ y_3 = m \cdot (x_1 - x_3) - y_1 \end{cases} \quad \text{with} \quad m = \frac{y_2 - y_1}{x_2 - x_1}$$

- **Point doubling.** The double of a point P_1 is $P_1 + P_1 = 2P_1 = (x_4, y_4)$ such that:

$$\begin{cases} x_4 = m^2 - 2x_1 \\ y_4 = m \cdot (x_1 - x_3) - y_1 \end{cases} \quad \text{with} \quad m = \frac{3x_1^2 + a}{2y_1}$$

Point addition (resp. point doubling) in affine coordinates requires 1 inversion, 2 multiplications, 1 squaring and 6 additions (resp. 1 inversion, 2 multiplications, 2 squarings and 8 additions). However, inversions in $\mathbb{F}_p$ are expensive in comparison to other field operations.

For the purpose of reducing the number of inversions required to perform the sequence of points additions (as required to compute a scalar multiplication), other coordinate systems can be used that we describe below.

2.2.4.2 Homogeneous projective coordinates

The homogeneous projective Weierstraß equation of $\mathcal{E}(\mathbb{F}_p)$ is:

$$Y^2Z = X^3 + aXZ^2 + bZ^3$$

It is obtained by using the mapping from homogeneous projective coordinates to affine coordinates:

$$(X, Y, Z) \mapsto (\frac{X}{Y}, \frac{Y}{Z}) \ , \ \ Z \neq 0$$

or the inverse mapping:

$$(x, y) \mapsto (\lambda x, \lambda y, \lambda) \ \ \forall \lambda \in \mathbb{F}_p^*$$

This transformation defines the equivalence relation below, and the corresponding equivalence class is denoted by $(X : Y : Z)$.

$$(X, Y, Z) \sim (\lambda X, \lambda Y, \lambda Z) \ , \ \forall \lambda \in \mathbb{F}_p^*$$

In the following let $P_1 = (X_1, Y_1, Z_1)$ and $P_2 = (X_2, Y_2, Z_2)$ such that $Z_1, Z_2 \neq 0$ and $P_1 \neq \pm P_2$. The homogeneous projective representation of O is $(0, \lambda, 0), \ \forall \lambda \in \mathbb{F}_p^*$.

- **Point opposite.** The opposite of a point $P_1 = (X_1, Y_1, Z_1)$ is $-P_1 = (X_1, -Y_1, Z_1)$.

- **Point addition.** The sum of two points $P_1 = (X_1, Y_1, Z_1)$ and $P_2 = (X_2, Y_2, Z_2)$ is a point $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$ such that:

$$\begin{cases} X_3 = BD \\ Y_3 = A(C - D) - B^3Y_1Z_2 \\ Z_3 = B^3Z_1Z_2 \end{cases} \quad \text{with} \quad \begin{array}{l} A = Y_2Z_1 - Y_1Z_2 \\ B = X_2Z_1 - X_1Z_2 \\ C = B^2X_1Z_2 \\ D = A^2Z_1Z_2 - B^3 - 2C \end{array}$$

- **Point doubling.** The double of a point P_1 is $2P_1 = (X_4, Y_4, Z_4)$ such that:

$$\begin{cases} X_4 = EB \\ Y_4 = A(D-E) - 2C^2 \\ Z_4 = B^3 \end{cases} \quad \text{with} \quad \begin{array}{l} A = 3X_1^2 + aZ_1^2 \\ B = 2Y_1Z_1 \\ C = BY_1 \\ D = 2CX_1 \\ E = A^2 - 2D \end{array}$$

In general, point addition (resp. point doubling) using homogeneous coordinates requires 12 multiplications, 2 squarings and 7 additions (resp. 7 multiplications, 5 squarings and 10 additions).

2.2.4.3 Jacobian projective coordinates

The Jacobian projective Weierstraß equation of $\mathcal{E}(\mathbb{F}_p)$ is:

$$Y^2 = X^3 + aXZ^4 + bZ^6$$

It is obtained by using the mapping from Jacobian projective coordinates to affine coordinates:

$$(X, Y, Z) \mapsto (\frac{X}{Z^2}, \frac{Y}{Z^3}) \ , \ Z \neq 0$$

or the inverse mapping:

$$(x, y) \mapsto (\lambda^2 x, \lambda^3 y, \lambda) \ , \ \forall \lambda \in \mathbb{F}_p^*$$

In this case, the equivalence class $(X : Y : Z)$ is defined by the relation:

$$(X, Y, Z) \sim (\lambda^2 X, \lambda^3 Y, \lambda Z) \ , \ \forall \lambda \in \mathbb{F}_p^*$$

The Jacobian projective representation of O is $(\lambda^2, \lambda^3, 0) \, \forall \lambda \in \mathbb{F}_p^*$.

- **Point opposite.** The opposite of a point $P_1 = (X_1, Y_1, Z_1)$ is $-P_1 = (X_1, -Y_1, Z_1)$.
- **Point addition.** The sum of two points $P_1 = (X_1, Y_1, Z_1)$ and $P_2 = (X_2, Y_2, Z_2)$ is $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$ such that:

$$\begin{cases} X_3 = D^2 - C^3 - 2AC^2 \\ Y_3 = D(AC^2 - X_3) - BC^3 \\ Z_3 = Z_1Z_2C \end{cases} \quad \text{with} \quad \begin{array}{l} A = X_1Z_2^2 \\ B = Y_1Z_2^3 \\ C = X_2Z_1^2 - A \\ D = Y_2Z_1^3 - B \end{array}$$

When P_1 and P_2 have equal Z coordinates (i.e. $Z_1 = Z_2 = Z$), then as shown by Meloni [119], the addition operation is simplified into:

$$\begin{cases} X_3 = D^2 - B - C \\ Y_3 = D(B - X_3) - Y_1(C - B) \\ Z_3 = Z(X_2 - X_1) \end{cases} \quad \text{with} \quad \begin{array}{l} A = (X_2 - X_1)^2 \\ B = X_1 A \\ C = X_2 A \\ D = Y_2 - Y_1 \end{array}$$

– **Point doubling.** The double of a point P_1 is $2P_1 = (X_4, Y_4, Z_4)$ such that:

$$\begin{cases} X_4 = C^2 - 2B \\ Y_4 = C(B - X_4) - 2A^2 \\ Z_4 = 2Y_1 Z_1 \end{cases} \quad \text{with} \quad \begin{array}{l} A = 2Y_1^2 \\ B = 2AX_1 \\ C = 3X_1^2 + aZ_1^4 \\ D = Y_2 - Y_1 \end{array}$$

Point addition (resp. doubling) in Jacobian coordinates requires 12 multiplications, 4 squarings and 7 additions (resp. 4 multiplications, 6 squarings and 11 additions). Overall, Jacobian coordinates provide faster point operations, and in particular for specific cases, e.g., the method presented by Meloni [119] for faster addition, that we detailed above. In addition, some scalar multiplication algorithms perform more point doubling than point addition operations, such as algorithms based on the non-adjacent form [91] or the windowed-non-adjacent form of the scalar [123], and hence favor Jacobian coordinates.

2.3 Side-channel attacks

A side channel is an unintended communication channel that leaks some information from a device through a physical observable, such as power consumption, execution time, electromagnetic radiation, or temperature. This source of leakage poses a threat if it is correlated with the internal state of the processing device, which is itself related to secret information.

2.3.1 Side-channel attack steps

In this Section, we describe the typical steps of a side-channel attack as formalized in publication 4 [7].

1. **Measurements and prepossessing.** During the first step of an attack, the adversary acquires observations of the side-channel leakage. A single measurement corresponding to one execution of the cryptographic function is called a *trace*. A trace is usually represented as a vector (e.g., $\mathbf{l}$) of sampled values of the side channel

indexed by time. One element of this vector is called a *sample* or *time sample* (the sample at time index t is $\mathbf{l}[t]$). Side-channel measurements are noisy physical observations. Signal processing techniques (e.g., amplitude demodulation) can be used to improve the quality of the traces.

2. **Leakage detection and mapping.** Leakage detection refers to any technique used to detect the presence of any secret-dependent information in the leakage traces. Leakage mapping extends the detection further, to pinpoint specific time samples, called *Point(s) Of Interest* (POI), that correspond to data or intermediate values that an attacker can subsequently use to mount an attack.

3. **Leakage exploitation.** The term leakage exploitation encompasses all techniques that exploit the leakage detected at the previous step to recover the secret key. We provide a taxonomy and examples of the main leakage exploitation approaches in Section 2.3.3, since this is the attack step we mainly study in this thesis. Leakage exploitation typically includes three steps:

   (a) Modeling (optional). With the possibility to acquire traces on the target device (or a clone) with controlled or random known inputs, this additional trace set can be used to model the side-channel behavior of the device. Put differently, to learn the relation between the processed data and the side-channel leakage.

   (b) Information extraction. In this step, the attacker uses the previously learned model or a hypothetical model to extract some information about the processed data.

   (c) Information processing. During the last step, the attacker combines the information extracted from different parts of the leakage, either from a single trace or over multiple traces, to recover the full secret key.

2.3.2 Leakage detection and mapping

In the following, we mention a few relevant examples for leakage detection and POI selection.

Test Vector Leakage Assessment. TVLA is a leakage detection methodology proposed by Goodwill et al. [69]. It is often used in evaluations [129] since it focuses on "fast" leakage detection and ignores

"time-consuming" leakage exploitation. TVLA aims to detect differences between two sets of observations given different data. The intuition behind it, is that there is data-dependent leakage if different intermediate values can be distinguished from the means of their leakage distributions. The TVLA methodology is based on a random vs. fixed Welch's t-test [167] and we detail its statistical process in the following. Let $\mathcal{A}$ and $\mathcal{B}$ be two sets of side-channel observations measured with random inputs and fixed input respectively. The first step of the test is to compute the following statistic:

$$s = \frac{\mathbb{E}(\mathcal{A}) - \mathbb{E}(\mathcal{B})}{\sqrt{\frac{\mathbb{V}(\mathcal{A})}{n_{\mathcal{A}}} + \frac{\mathbb{V}(\mathcal{B})}{n_{\mathcal{B}}}}}$$

where $n_{\mathcal{A}}$ and $n_{\mathcal{B}}$ are the sizes of the sets $\mathcal{A}$ and $\mathcal{B}$ respectively. The null hypothesis H_{null}, that the means of the two sets are indistinguishable (i.e. $H_{null} : \mathbb{E}(\mathcal{A}) = \mathbb{E}(\mathcal{B})$), is rejected at significance level α if $|s| > t_{\frac{\alpha}{2},df}$, with $t_{\frac{\alpha}{2}df}$ being the value of the Student's t-distribution with df degrees of freedom:

$$df = \frac{(\frac{\mathbb{V}(\mathcal{A})}{n_{\mathcal{A}}} + \frac{\mathbb{V}(\mathcal{B})}{n_{\mathcal{B}}})^2}{\frac{\mathbb{V}(\mathcal{A})^2}{n_{\mathcal{A}}^2(n_{\mathcal{A}}-1)} + \frac{\mathbb{V}(\mathcal{B})^2}{n_{\mathcal{B}}^2(n_{\mathcal{B}}-1)}}$$

Regarding guidelines and best practices on using and interpreting results from leakage detection tests such as TVLA, we refer to the comprehensive works in [153, 168, 169].

Correlation-based leakage detection and POI selection. Durvaux and Standaert describe an alternative leakage detection test based on Pearson's correlation coefficient (ρ), called the ρ-test [56]. In this case, the statistic to compute is related to an intermediate variable X and corresponding leakage observation set $\mathcal{L}_X$, and is the following:

$$s = \rho(\mathcal{L}_X, \hat{\mathsf{m}}(X))$$

where $\hat{\mathsf{m}}$ is an estimated side-channel leakage model or a hypothetical leakage function, e.g., the HW function. The next step is to apply Fisher's z-transformation and to normalize by $\frac{1}{\sqrt{n_{\mathcal{L}_X}-3}}$, with $n_{\mathcal{L}_X}$ being the number of observations in $\mathcal{L}_X$, to obtain:

$$s_z = \frac{1}{2 \cdot \sqrt{n_{\mathcal{L}_X} - 3}} \cdot \ln\left(\frac{1+s}{1-s}\right)$$

In [56] a threshold of 5 is used, but it can also be calculated based on a chosen significance level since s_z can be approximated by a sample

distributed according to $\mathcal{N}(0,1)$. Since this leakage detection test is based on a specific intermediate X, it can also be used to pinpoint relevant POI for exploitation.

2.3.3 Side-channel leakage exploitation

Leakage exploitation techniques can be categorized in two families: unprofiled and profiled attacks.

Unprofiled attacks. this kind of exploitation does not require a preliminary modeling phase and potentially assumes a hypothetical leakage model. We describe in the following a few main examples of unprofiled attacks.

1. **Simple analysis.** In this kind of attack, only a single trace is needed and the secret information can be directly "read" from the side-channel trace. Condition branching that depends on the secret is a source of leakage that can be exploited using simple analysis. For instance, the Double-and-Add ECSM described in Algorithm 3 exhibits distinguishable patterns on the side-channel trace that depend on the bits of k (when a bit is 1 a point doubling is always followed by a point addition).

2. **Differential analysis.** Differential analysis as opposed to simple analysis, typically requires the acquisition of many traces $(l_i)_{0\leq i<N}$ for the secret key with known inputs (e.g., the AES plaintext). Then, by making a guess k^* on a part of the key k (e.g., an AES key byte), it is possible to compute hypothetical values $(v_i^{k^*})_{0\leq i<N}$ of some intermediate that depends on k and the input (e.g., the AES Sbox output). In the following, we describe a few techniques for differential key recovery (for conciseness we use the term key to designate the targeted part of the full key):

   (a) Differential Power Analysis (DPA). First proposed by Kocher et al. [100], in a DPA the attacker can partition the traces $(l_i)_{0\leq i<N}$ into two sets $\mathcal{S}_0^{k^*}$ and $\mathcal{S}_1^{k^*}$ for each key candidate k^* based on one bit of the hypothetical intermediates $(v_i^{k^*})_{0\leq i<N}$. For each key candidate this partitioning is different, and the leakage model assumed here is that the side-channel behavior of a bit set to 0 is different from a bit set to 1. The next step of the attack is to compute the differences $\Delta^{k*} = \mathbb{E}(\mathcal{S}_0^{k^*}) - \mathbb{E}(\mathcal{S}_1^{k^*})$. This difference or any measure used to distinguish or score key candidates is called a *distinguisher*

and its values are called *scores*. For instance, in this case, the distinguisher is the difference of means. If enough traces are used and the target implementation is unprotected, Δ^{k^*} exhibits peaks only for the correct key candidate $k^* = k$.

(b) Correlation Power Analysis (CPA). This attack was introduced by Brier, Clavier and Olivier [25] and assumes the Hamming Weight (HW) or the Hamming Distance (HD) leakage model. In the following, for simplicity we consider an example using the HW model. This time, the manipulation of intermediates is assumed to leak a linear function of their HW perturbed by some independent noise. To find the correct key candidate, Brier et al. use the Pearson correlation coefficient ρ as a distinguisher. An attacker will estimate for each key candidate k^* the following score:

$$\rho^{k^*} = \frac{\mathbb{C}ov((l_i)_{0\leq i<N}, (h_i^{k^*})_{0\leq i<N})}{\sqrt{\mathbb{V}((l_i)_{0\leq i<N})} \cdot \sqrt{\mathbb{V}((h_i^{k^*})_{0\leq i<N})}} \tag{2.1}$$

where $\mathbb{C}ov$ denotes the covariance operator and $(h_i^{k^*})_{0\leq i<N} = (\mathsf{HW}(v_i^{k^*}))_{0\leq i<N}$. The key candidate selected is the one maximizing the absolute value of the correlation coefficient.

(c) Linear Regression Analysis (LRA). Schindler et al. describe an efficient method for side-channel modeling/profiling [145] based on a linear model. Doget et al. later extend this work to unprofiled key recovery [51]. Specifically, linear regression models the relation between a variable X and an observed corresponding leakage variable L, using a linear function of the $|X| = n$ bits of X according to a basis $\mathbf{b} = [\beta_0, \beta_1, \ldots, \beta_n]$ by:

$$l = \beta_0 \cdot x_0 + \ldots + \beta_{n-1} \cdot x_{n-1} + \beta_n$$

with x_i being the i-th bit of x. Given a set of m leakage observations $\mathbf{l}^k = [l_0^k, l_1^k, \ldots l_{m-1}^k]$ corresponding to an intermediate that depends on a part k of the secret key (e.g., the Sbox output), an unprofiled LRA proceeds as follows. First, the sets of hypothetical values for the considered intermediate are computed for each key candidate k^*, that we denote by $\mathcal{X}_{k^*}$. Then, the vectors of coefficients $\mathbf{b}^{k^*}$ are estimated for each k^* as:

$$\mathbf{b}^{k^*} = (\mathbf{X}_{k^*}^{\mathrm{T}} \cdot \mathbf{X}_{k^*}) \cdot \mathbf{X}_{k^*}^{\mathrm{T}} \cdot \mathbf{l}^k$$

with $\mathbf{X}_{k^*}$ a matrix whose lines are the binary representations of the values in $\mathcal{X}_{k^*}$ appended by one (to estimate the coefficients $\beta_n^{k^*}$). Then, each candidate is scored based on how well its estimated model $\mathbf{b}^{k^*}$ fits with the observations in $\mathbf{l}^k$. To do so, the distinguisher used in [51] is the coefficient of determination R^2, which is defined as:

$$R^2_{k^*} = 1 - \frac{\mathbb{E}\left((\mathbf{l}^k - \mathbf{X}_{k^*} \cdot \mathbf{b}^{k^*})^2\right)}{\mathbb{V}(\mathbf{l}^k)}$$

$R^2 \in [0, 1]$ and 1 corresponds to the best fitting model.

(d) Mutual Information Analysis (MIA). Another example of differential attacks are MIA introduced by Gierlichs et al. [66]. As opposed to classical differential attacks such as the original DPA or CPA, MIA does not require a hypothetical leakage model and uses the Mutual Information (MI) of the leakage and the intermediate value as a distinguisher. Formally, the key is selected as the one maximizing the value of the information theoretic measure $\mathsf{MI}((l_i)_{0 \leq i < N}; (v_i^{k^*})_{0 \leq i < N})$. We detail in Section 2.3.5 the computation of the MI since it is also used as a security evaluation metric.

Profiled attacks. profiled attacks are more powerful than unprofiled attacks, since as opposed to the latter, they do not rely on a hypothetical model, but aim to approximate the real leakage behavior of the target as closely as possible. However, a profiled attack requires an additional set of traces with known inputs on the same target or a clone device in order to model the leakage. Access to such a profiling set is not always possible.

1. **Template attack.** Template attacks are one of the most notable examples of profiled attacks and were introduced by Chari, Rao and Rohatgi [34]. They model the side-channel leakage as a random sample drawn from a Gaussian probability distribution. Formally, the leakage l of an intermediate v is assumed to follow a distribution $\mathcal{N}(\mu_v, \sigma^2)$, where the mean μ_v depends on the target value v, and the noise variance σ^2 can either be assumed to be independent from v for simplicity or estimated for each value v. When multiple POI are taken into account, the distribution is multivariate. The profiling phase of a Template attack entails the estimation of the mean and the variance, or covariance matrix when considering multivariate leakages, for each possible value of the target variable. During the attack phase, leakages are classified

based on a maximum-likelihood criteria. Formally, the probability of observing a leakage l if the intermediate V is equal to v is estimated as:

$$\Pr[l|V = v] \propto \mathcal{N}(l|\mu_v, \sigma^2) \tag{2.2}$$

Template attacks can be used to exploit the direct leakage of parts of a secret key, but also to exploit the leakage of intermediates. Since these intermediates depend on the key, using the input knowledge their recovered information can be combined to recover the key.

2. **Linear regression attack.** A variation of Template attacks are linear-regression-based attacks. They were proposed by Schindler et al. [145] and can be used to model leakages similarly to a Template attack. We previously described the unprofiled version of LRA as suggested by Doget et al. [51]. We do not detail the profiled version of LRA further in this section as we provide a practical example of a linear-regression-based profiled attack in Section 3.2.

3. **Machine learning and deep learning.** In the recent years, the side-channel community gained interest in machine learning and deep learning methods. In particular, regarding machine learning, techniques such as self organizing maps, support vector machines and random forests were compared against classic Template attacks and were shown to improve the attack outcome [85, 107]. The application of deep neural networks for side-channel analysis was originally proposed by Gilmore, Hanley and O'Neill, and were applied to masked AES implementations [67]. Since then a multitude of works have followed, improving the understanding of neural networks in the field of side-channel analysis [115, 111, 31].

D&C, E&P and analytical/algebraic attacks. In addition to the previous taxonomy, an attack can be classified into different categories based on the strategy it follows to extract a secret key. This strategy usually depends on the target implementation and includes the following approaches: D&C, E&P and algebraic/analytical methods. In a D&C attack key words are recovered independently. This is the usual modus operandi of side-channel attacks on block ciphers such as AES. In an E&P attack, key words are recovered recursively. This strategy is potentially useful for asymmetric cryptography due to the recursive nature of computing a modular exponentiation for RSA or a scalar multiplication for ECC.

Algebraic [137] or analytical [163] strategies are more advanced and allow to exploit all the relations between the intermediate values and

the key words. The main example of analytical attacks are Soft Analytical Side-Channel Attacks and were introduced by Veyrat-Charvillon, Gérard and Standaert at Asiacrypt 2014 [163]. In this thesis, we take a particular interest in these methods and describe them further in Section 2.5.

Vertical, Horizontal and HO attacks. Along with the previous classifications, a side-channel attack can be furthermore labeled according to the segments or information from the traces that it exploits. First, vertical analysis typically designates classic side-channel attacks that exploit the same time sample or a few time samples corresponding to the same time moment or operation across multiple side-channel traces. Two main examples of vertical attacks are classic DPA [100] and CPA [25]. As opposed to vertical analysis, horizontal analysis, and in particular in the asymmetric cryptography context, refers to attacks using multiple samples per operation and exploiting multiple target operations or intermediates. The term multivariate is sometimes used interchangeably with horizontal since their definitions are identical. Both terms do not exclude the use of multiple side-channel traces. The main examples of horizontal/multivariate attacks are the seminal Big Mac attack of Walter [166] and the more recent SASCA [163].

The last class of analysis that we mention in this taxonomy are High-Order (HO) attacks, which are commonly used to break protected implementations that employ the masking countermeasure (which is detailed in Section 2.4.1.1). HO attacks exploit the leakage of all shares jointly to recover the shared secret. However, the main difference is that HO attacks imply the estimation of a higher-order statistical moment of the leakage distribution, e.g., when a secret variable is shared into d shares, the information that can be recovered on it from the side-channel leakage lies in the d^{th} order moment of the leakage distribution. A HO attack requires a single leakage point when the shares are handled in parallel, and it can be considered as multivariate if the leakage of the shares is spread across multiple time samples, e.g., when the shares are handled sequentially.

2.3.4 Side-channel attacks on ECC

Due to the efficiency of elliptic curve based cryptosystems in comparison to other public-key cryptosystems such as RSA, they have been widely deployed in modern information systems and in particular on embedded devices, and thus are subject to side-channel attacks. As previously mentioned, the main operation in elliptic curve systems is the ECSM, therefore it is also the primary attack target. In the following, we provide

a brief survey of side-channel attacks on ECSM implementations.

First, SPA [99] targets insecure scalar multiplication algorithms and exploits the fact that the sequence of operations executed depends on the secret scalar. An example of such an algorithm is the Double-and-Add scalar multiplication previously described in Algorithm 3. A straightforward countermeasure to these attacks is the use of algorithms with a regular execution flow, such as the Montgomery ladder [93] described in Algorithm 5.

Next, Coron has shown that a classic DPA, such as described in Section 2.3.3, can recover the secret scalar using multiple side-channel traces [43]. Differential attacks are not applicable to ECDSA since the scalar nonces are never reused. In the case of ECDH, this attack can be prevented by using blinding/randomization countermeasures [43]. This kind of countermeasure and all following mentioned countermeasures will be detailed in Section 2.4.

Template attacks have also been used to break ECC implementations. For instance, Medwed and Oswald use a single trace to compromise an ECDSA implementation [118]. Batina et al. then proposed Online Template Attacks (OTA) to reduce the number of necessary profiling/templating traces [12]. Another example of advanced yet practical template attack on ECC implementations is the work of Dugardin et al. [54], which combines both horizontal and vertical leakage. Following, Järvinen and Balasch build upon template attacks by targeting scalar multiplications with precomputed points [90]. Attacks that only require a single trace additionally include horizontal correlation analysis, which was proposed by Clavier et al. [39]. Both template attacks and horizontal correlation attacks rely on the knowledge of the input point and can be thwarted by point randomization, point blinding [43], random elliptic curve or field isomorphisms [92] or field arithmetic blinding [151, 55].

Horizontal collision attacks [14], take advantage of the observation that for a certain scalar bit value, identical intermediate values are manipulated at different instants of the execution of the scalar multiplication. If these collisions can be detected then information can be recovered on the secret scalar. The idea of horizontal collision attacks comes from the seminal work of Walter [166] targeting sliding windows implementations [171]. Another example of collision attacks is the doubling attack of Fouque and Valette [61]. This attack is a chosen input attack and is based on the fact that an adversary can detect when the same operation is performed twice. Other examples of recent collision-based attacks include the work of Homma et al. [84] and the work of Hanley et al. [79]. Collision-based attacks are very powerful since they sometimes

do not require any profiling or even known inputs. These attacks can be hindered by the shuffling countermeasure [106] (in general the shuffling countermeasure makes all the previous attacks more challenging depending on the side-channel noise) or randomization techniques [92, 151, 55].

After a side-channel attack on an ECC implementation, in the case of ECDH, computational power can be used to mitigate a possible lack of information using enumeration [105]. Enumeration combines the probabilistic information on different parts of the secret scalar to recover its full value. For ECDSA, a helpful strategy is to partially attack the random scalar nonces, recover their first few bits, and then apply lattice cryptanalysis in order to recover the signer's secret key, as shown by Nguyen and Shparlinski [128].

2.3.5 Side-channel evaluation metrics

When it comes to side-channel security evaluations, it is possible to rely on some attack metrics, for instance, to estimate the remaining computational effort required to recover a secret key after an attack. In practice, this is captured by information theory or security metrics, that cater to the realistic case where an adversary performs offline enumeration to recover a full secret key.

In this subsection, we define some of these metrics based on the work of Standaert, Malkin and Yung [154] and the work of Köpf and Basin [102]. As previously explained, the outcome of an attack aiming to recover a secret key word k is usually a vector of scores that we denote by $\mathbf{scores}^k$ (the superscript k refers to the fact that we exploit traces coming from the execution of the cryptographic function parametrized by the secret k). These scores, for each key candidate k^*, can be probabilities, e.g., after a Template attack, or correlation coefficients after a CPA. Since classic side-channel attacks usually follow a D&C strategy and recover these scores for each word of the secret independently, the following metrics are defined for a key word k. Their definition can be naturally extended to the full key, but their estimation becomes more elaborate in that case. Additionally, since these metrics are estimated experimentally, their values are averaged across multiple experiments/attacks for many possibly random keys and multiple sets of corresponding side-channel leakages, to provide a precise estimation. In the following, $\mathbb{E}$ denotes the expected value or mean operator over these experiments/attacks.

2.3.5.1 Guessing entropy

The Guessing Entropy (GE) is calculated as the average Rank of the correct key k, where the Rank is defined as the position of k in the sorted vector of scores, with the position count starting at 1. Concretely, the GE measures the average number of candidates to test after an attack to recover the correct key. It is defined as:

$$\mathsf{GE} = \mathbb{E}[\mathsf{Rank}(k, \mathbf{scores}^k)] \tag{2.3}$$

2.3.5.2 Success rate

The Success Rate (SR) of order o (starting at 1) is the probability that the correct key is ranked among the first o candidates in the sorted vector of scores. We define the SR of order 1, as it is the most relevant to the rest of this thesis and simply denote it as SR. It is given by:

$$\mathsf{SR} = \Pr[\mathsf{Rank}(k, \mathbf{scores}^k) = 1] \tag{2.4}$$

2.3.5.3 Mutual information

The mutual information is an Information Theoretic (IT) metric and as opposed to the other metrics it requires the scores to be probabilities. For instance, the probabilities provided by a profiled attack (e.g., a Template attack) correspond to the term $\Pr[l|v]$ in the equation below that defines the MI. $\Pr[l|v]$ is the probability of observing a leakage l when the manipulated value is v. The MI measures the information in bits recovered on the target variable from the side-channel leakage and is estimated by sampling using n^v sampled leakages for each value v according to:

$$\mathsf{MI}(V; L) = \mathrm{H}(V) - \sum_v \Pr[v] \cdot \frac{1}{n^v} \sum_l \log_2 \Pr[v|l] \tag{2.5}$$

The previous definition assumes either a perfect modeling with the ability to estimate accurately the true leakage distribution, or a simulated case where the leakage distribution is known. However, in practical cases the true leakage distribution is in fact unknown and the estimation of the true MI is not possible. Based on this observation, Renauld et al. [138] introduce the Perceived Information metric (PI) that captures the information conveyed by the leakage assuming the approximated leakage distribution that we denote by $\tilde{\Pr}[v|l]$:

$$\mathsf{PI}(V; L) = \mathrm{H}(V) - \sum_v \Pr[v] \cdot \frac{1}{n^v} \sum_l \log_2 \tilde{\Pr}[v|l] \tag{2.6}$$

2.4 Side-channel attack countermeasures

In the following, we describe the most common countermeasures against side-channel attacks to protect symmetric and asymmetric cryptography implementations. In particular, we introduce the two most popular approaches to protect block ciphers such as AES, i.e. masking and shuffling. Then, we explain popular countermeasures for ECSM executions, which mainly consist in some kind of randomization of the inputs or intermediate values.

2.4.1 Block ciphers

2.4.1.1 Masking

The masking countermeasure proposed by Ishai, Sahai and Wagner [88] aims at randomizing the processed data such that the sensitive data is never explicitly processed by the circuit. It works by decomposing all intermediate variables into d secret shares. This secret-sharing-based countermeasure was first introduced by Chari et al., Goubin and Patarin, and Messerges [33, 71, 120]. Particularly, Chari et al. showed that for sufficient side-channel noise, the security of the implementation, i.e. the number of observations required to recover the secret, grows exponentially with the security order d.

One prevalent scheme is Boolean masking. Concretely, a $(d-1)$-th order secure Boolean masking scheme splits a sensitive value x into d shares $(x_0, x_1, \ldots, x_{d-1})$, where the $(d-1)$ first shares are randomly generated and the d-th share is computed such that $x = x_0 \oplus x_1 \oplus \cdots \oplus x_{d-1}$. This implies that with strictly less than d shares, the adversary does not obtain any information on x. The cost of applying Boolean masking to linear operations is linear in d, whereas for non-linear operations the cost is quadratic in d.

2.4.1.2 Shuffling

The shuffling countermeasure, originally proposed by Herbst, Oswald and Mangard [83], consists in randomizing the time location of the processing of sensitive intermediate values. In practice, at each execution of the cipher, a random permutation of size π is generated and the order of identical operations or blocks of operations is based on the generated permutation. Broadly speaking, shuffling scatters the useful information over π distinct time samples across all traces.

The security impact of this countermeasure is linear in the number of shuffled operations π and the attacker's ability to distinguish the target

operation among the permuted set [38, 83]. Regarding the cost of this countermeasure, first, the overhead of shuffling operations is generally assumed to be linear but highly depends on the device [164]. Second, shuffling requires the generation of a permutation over π elements. In this respect, Veyrat-Charvillon et al. [164] suggest to use a biased version of the Knuth (or Fisher–Yates) shuffle algorithm [97] that requires $\pi \cdot \log_2(\pi)$ random bits to generate a permutation of size π.

2.4.2 Elliptic curve cryptography

2.4.2.1 Scalar blinding

This countermeasure consists in hiding the secret scalar by adding a random multiple of the elliptic curve order at each new execution of the ECSM. The secret scalar k is randomized as $k' = k + r \cdot \text{ord}(\mathcal{E}(\mathbb{F}_p))$ with r a random scalar of a chosen size. The calculation of the ECSM with k' as input leads to the expected result since $k' \cdot P = (k + r \cdot \text{ord}(\mathcal{E}(\mathbb{F}_p))) \cdot P = k \cdot P + r \cdot \text{ord}(\mathcal{E}(\mathbb{F}_p)) \cdot P = k \cdot P + r \cdot O = k \cdot P$. The overhead of scalar blinding is approximately $\frac{p}{|r|}$ of the ECSM's cost. Another variant of scalar blinding, is called euclidean blinding or scalar splitting and was introduced by Ciet and Joye [36]. It consists in writing $a = \lfloor k/r \rfloor$ and $b = k\%r$ for a random r, then the scalar multiplication becomes $k \cdot P = r \cdot (a \cdot P) + b \cdot P$.

This countermeasure comes however with a cautionary note relating to the requirements on the size of the blinding parameter r. Schindler et al. have shown in many articles including [144, 146, 147] that scalar blinding is not as resistant as expected. In particular, the attacks they present are able to recover the secret by observing many of its randomizations when the size of r is not large enough. This is particularly the case for pseudo-Mersenne primes used by the NIST ECC standards [130].

2.4.2.2 Point randomization

As presented by Coron [43], using a projective coordinate system, the input point to the ECSM can be randomized such that its representation is unknown to an attacker. To randomize a point $P = (x, y) \in \mathcal{E}(\mathbb{F}_p)$, a scalar $\lambda \in \mathbb{F}_p^*$ is generated randomly and the point is represented in homogeneous projective coordinates as $(\lambda \cdot x, \lambda \cdot y, \lambda)$ (hence requiring 2 field multiplications), or in jacobian projective coordinates as $(\lambda^2 \cdot x, \lambda^3 \cdot y, \lambda)$ (requiring 3 multiplications and 1 squaring). We analyze both the theoretical and the practical security of this countermeasure in Chapter 4.

2.4.2.3 Point blinding

Point blinding is another countermeasure also proposed by Coron [43], which hides the input point from an attacker such that differential attacks or any attack that relies on the knowledge of the input point is hindered. It consists in masking the input point with a randomly selected point. Concretely, first a random point $R \in \mathcal{E}(\mathbb{F}_p)$ is generated for which $S = k \cdot R$ is known. Then, the ECSM computes $k \cdot (P + R)$ instead of $k \cdot P$. At the end, S is subtracted from the output of the ECSM to yield the expected result since $k \cdot (P+R) - S = k \cdot P + k \cdot R - k \cdot R = k \cdot P$.

This countermeasure is only efficient in the case of ECDH when the secret key is reused for multiple ECSMs, and the pair $(R,\ S)$ is efficiently refreshed by generating a random bit b and replacing R and S by $(-1)^b \cdot 2 \cdot R$ and $(-1)^b \cdot 2 \cdot S$ (or more generally $\alpha \cdot R$ and $\alpha \cdot S$ for a small α). The overhead of this countermeasure involves the cost of 2 point additions to add R before the ECSM and subtract S after, and the cost of 2 point doublings to refresh R and S.

2.4.2.4 Random curve/field isomorphism

The main example of this kind of countermeasure consists in using a random curve isomorphism at each ECSM execution. This countermeasure was proposed by Joye and Tymen [92]. An elliptic curve isomorphism is defined as:

Curve isomorphism. Let $\mathcal{E}_1(\mathbb{F}_p) : y^2 = x^3 + a_1 x + b_1$ and $\mathcal{E}_2(\mathbb{F}_p) : y^2 = x^3 + a_2 x + b_2$ be two elliptic curves over $\mathbb{F}_p$. $\mathcal{E}_1$ and $\mathcal{E}_2$ are isomorphic if $\exists v \in \mathbb{F}_p^*$ such that $a_1 = v^4 a_2$ and $b_1 = v^6 b_2$. And the corresponding isomorphism is:

$$\mathcal{E}_1(\mathbb{F}_p) \to \mathcal{E}_2(\mathbb{F}_p)$$
$$(x, y) \mapsto (v^2 x, v^3 y)$$

This countermeasure then consists in generating a random $v \in \mathbb{F}_p^*$, then translating $\mathcal{E}(\mathbb{F}_p)$ into the corresponding isomorphic curve and its cost is similar to the one of projective coordinate randomization. Joye and Tymen also suggest to follow the same approach to randomize the underlying field of the elliptic curve and in particular on extension fields [92].

2.4.2.5 Field arithmetic blinding

This countermeasure is based on a redundant representation of the field elements and was proposed by Smart, Oswald and Page [151], and later developed in [55]. This approach allows to hide intermediate values

using random multiples of p. This representation can be refreshed but intermediate values are not fully reduced modulo p but instead modulo $c \cdot p$ with c coprime to p. In general, the overhead depends on $|c|$, since all multi-precision operations are no longer performed on $|p|$ bits but instead on $|c| + |p|$ bits.

2.5 Soft analytical side-channel attacks

We have previously pointed out in Section 2.3 that attacks can follow different strategies to exploit side-channel information, ranging from simple approaches such as D&C or E&P strategies, to more complex algebraic or analytical strategies. In this thesis, we are particularly interested in the latter more advanced methods on account of their worst-case potential. The main example of such advanced methods that are best suited for side-channel analysis are Soft-Analytical Side-Channel Attacks. These attacks were introduced by Veyrat-Charvillon, Gérard and Standaert [163] as a new approach that combines the noise tolerance of classic differential attacks with the optimal data complexity of algebraic attacks [137]. SASCA exploit a larger number of intermediates in comparison to standard attacks and work in three steps. First, they build a large graphical model, called a factor graph, containing the intermediate variables linked by constraints corresponding to the operations executed during the target algorithm. This is analogous to a cipher's system of equations that algebraic attacks aim to solve. Then, they extract the distributions of the intermediate values from the leakage traces. Finally, they propagate the side-channel information throughout the factor graph using the Belief Propagation (BP) algorithm [132], to find the marginal distribution of the secret key, given the distributions of all intermediate variables.

In the following, we define factor graphs and provide examples on how to construct them. Then, we describe the BP algorithm.

2.5.1 Factor graph

Formally, a factor graph represents the factorization of a function. It is bipartite and contains two kinds of nodes: variable and factor nodes. In the context of side-channel attacks, this function is the probability distribution of the secret key conditioned by the leakage of the intermediate values of the cipher. The variable nodes in the graph are the intermediates of the cipher. The factor nodes are the functions of which the target key distribution is a product of. A factor is connected by an

edge to all the variables it depends on. When depicting a factor graph, variable nodes are represented by circles and factor nodes by squares or rectangles.

How a factor graph is built and how the factors are defined are better understood through examples. We consider a simple key byte addition (XOR) and SubBytes operation given by $z = \mathsf{S}(x \oplus k)$ (e.g., as performed in the AES). The corresponding factor graph is given in Figure 2.3. The factors $f_\oplus$ and f_S encode the correctness of the variable values given a defined relation or equation. In this case, $x \oplus k = y$ and $\mathsf{S}(y) = z$, respectively. In general, any factor f that depends on a set of variables returns 1 if and only if the values given to the set of variables verifies the relations encoded by f. For instance, $f_\oplus$ and f_S are defined as:

$$f_\oplus(x,k,y) = \begin{cases} 1 \ if \ \ x \oplus k = y, \\ 0 \ otherwise. \end{cases} \qquad f_\mathsf{S}(y,z) = \begin{cases} 1 \ if \ \ \mathsf{S}(y) = z, \\ 0 \ otherwise. \end{cases}$$

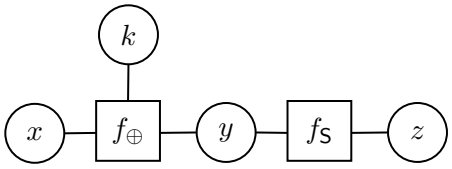


Figure 2.3: XOR and SubBytes factor graph example.

The side-channel information is injected in the graph by connecting every leaking variable to a corresponding leakage factor. This leakage factor is a leaf node and returns a probability vector on the variable given its side-channel leakage.

2.5.2 Belief propagation

The belief propagation algorithm is a message-passing algorithm for inference on graphical models such as factor graphs. It is also known as the sum-product algorithm. The belief propagation algorithm allows to efficiently compute the exact marginal distribution of a variable in the graph if it is tree-like. We follow the description of the BP algorithm given by MacKay [110].

We denote by $\mathbf{x}$ the set of N variables in the graph $\{x_n\}_{n=1}^N$. The m^{th} factor is denoted by f_m and the set of variables it depends on by $\mathbf{x}_m$. The set of variables $\mathbf{x}_m$ excluding variable x_n is denoted $\mathbf{x}_m|_n$. We refer by $\mathcal{N}(m)$ to the set of indices of $\mathbf{x}_m$, and by $\mathcal{N}(m)|_n$ to the set $\mathcal{N}(m)$ excluding index n. We also denote the set of factor indices in

which a variable x_n is involved by $\mathcal{M}(n)$, and the set of factors in which a variable x_n is involved excluding f_m by $\mathcal{M}(n)|_m$. The belief propagation algorithm works by passing two different types of messages on the edges of the graph for every pair of variable and factor (x_n, f_m) if related. This is verified if $x_n \in \mathbf{x}_m$ and $m \in \mathcal{M}(n)$:

From x_n to f_m: $$q_{n\to m}(x_n) = \prod_{m' \in \mathcal{M}(n)|_m} r_{m'\to n}(x_n).$$

From f_m to x_n: $$r_{m\to n}(x_n) = \sum_{\mathbf{x}_m|_n} \Big(f_m(\mathbf{x}_m) \prod_{n' \in \mathcal{N}(m)|_n} q_{n'\to m}(x'_n) \Big).$$

At first, all messages from variables to factors are initialized to the variables' a priori probabilities. These probabilities can either be passed from the leakage factors or set to one assuming uniformity. Messages from factors to variables are also initialized to one. Then, a node sends a message to a neighbor based on all the messages received from its other neighbors and following the previously described belief propagation rules. One iteration of the BP algorithm is reached once all these messages have been sent. The message passing repeats until a convergence condition is reached, for instance, a maximum number of iterations or a threshold for a statistical distance from the previous to the current iteration. If the graph is tree-like and does not contain any cycles, the number of steps required to get all messages to converge is equal to the longest path in the graph (the diameter). Once convergence occurs, the marginal distribution of a variable x_n can be recovered by multiplying together all its incoming messages: $\prod_{m \in \mathcal{M}(n)} r_{m\to n}(x_n)$.

The BP algorithm returns the exact marginals of the variables when the graph is a tree [132]. When the graph contains cycles, the BP algorithm can still be applied by initializing the messages and iterating the message-passing multiple times until the convergence condition is fulfilled. This method is called loopy belief propagation and usually provides a good approximation of the marginals.

The complexity of running the BP algorithm on a factor graph depends on the number of iterations required for convergence (which relates to the size of the graph), the size of the variables, and the number of variables connected to the factors. We refer to the number of variables on which a factor f depends by degree and denote it by $deg(f)$. The running time of the BP algorithm is dominated by the factor to variable message passing. The straightforward computation of a message from a factor f to any of its related variables requires $(2^{vs})^{def(f)}$ operations, with vs the variables' size.

2.5.3 Local random probing model

Based on the high time complexity of BP-based attacks, Guo et al. have proposed a short-cut [78]. It is a new model to reason about the security of cryptographic implementations against worst-case attacks and was named the Local Random Probing Model (LPRM) since it utilizes the Random Probing Model (RPM) [52]. Concretely, the LRPM gives an upper bound on the information that can be obtained by decoding the factor graph using BP without running the BP algorithm.

To model the side-channel security of an implementation in the LRPM, the first step is to construct its factor graph, as done when performing a BP-based attack. Then, instead of injecting in the graph knowledge about the intermediates through a probability distribution extracted from the leakage, only the amount of information (MI) on the intermediates is modeled with the RPM. Then, instead of propagating probability distributions, local information propagation rules are used to spread the MI values through the graph. The output MI on an edge is estimated based on the input MI values on the other connected edges. These values are normalized by the variable size such that the information is at least 0 and at most 1. The rules used to calculate the MI passed along the edges of the graph are detailed in Figure 2.4. In a nutshell, information at factor nodes is multiplied, and at variable nodes it is summed. The first rule is a variation of the piling-up lemma as used in the RPM to model masking security. The last rule is a good approximation when the involved variables' distributions are not too correlated, based on the Independent Operations' Leakages assumption (IOL) introduced by Grosso and Standaert [77]. This bound can also be illustrated by the following example: for three random variables X, Y and Z, we have that $\mathsf{MI}(X; Y, Z) \leq \mathsf{MI}(X; Y) + \mathsf{MI}(X; Z)$ and the equality is achieved if Y and Z are independent.

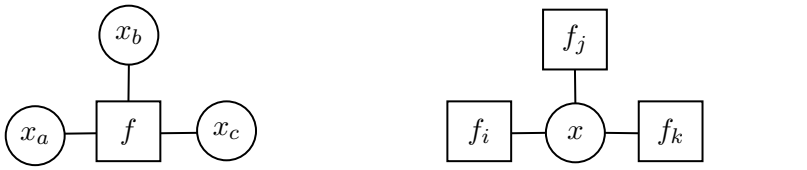


Figure 2.4: LRPM information propagation rules.

Chapter 3

Shortcut Formulas for Horizontal Attacks on ECC

In this chapter, we are interested in horizontal attacks following an E&P strategy against ECC implementations. In particular, Poussier, Zhou and Standaert, presented a systematic approach to perform such horizontal attacks against the Montgomery ladder ECSM [134]. This approach extracts a significant amount of information from a leakage trace by targeting multiple intermediate values. Due to its horizontal and nearly worst-case qualities, this attack is a relevant candidate for side-channel security evaluations. However, a drawback of highly horizontal attacks is that they usually rely on some knowledge of the implementation and most importantly on the modeling and exploitation of many leakage samples.

The work presented in this chapter is motivated by the last observation and additionally inspired by shortcut evaluation strategies considered for implementation security in symmetric cryptography. Shortcut formulas are defined as techniques to efficiently predict the outcome of side-channel attacks and were proposed, extended and investigated in various papers [139, 59, 48, 109, 58, 77]. Still, such techniques were never applied before for the side-channel analysis of asymmetric cryptography implementations. In the context of nearly worst-case horizontal attacks, we derive efficient shortcut formulas to estimate their success rate against ECSM implementations. This estimation is a function of the number of considered samples and the noise level, and similarly to related works on symmetric cryptography relies on a set of leakage assumptions.

The rest of this chapter is organized as follows. First, Section 3.1 describes the target ECC implementation example we study in this chapter. Then, Section 3.2 introduces the systematic Horizontal Differential

Power Analysis (HDPA) of Poussier et al. [134]. Next, Section 3.3 explains the rationale behind our approach and the goal of our research. Section 3.4 details the efficient derivation of the success rate and describes its underlying assumptions on the leakage, which are later discussed and validated. Finally, Section 3.5 reports results from simulated experiments and Section 3.6 provides experimental confirmation that the proposed tools lead to good predictions of the attacks' success on a real case study.

3.1 ECC implementation case-study

In the following sections, we examine the same reference ECSM analyzed in [134], which operates on the NIST P-256 curve defined over a prime field $\mathbb{F}_p$ with $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ [130]. The target considered is a 32-bit device and the ECSM is implemented using a regular Montgomery ladder as described in Algorithm 6. This ECSM processes the $l = 256$ bits of k iteratively from most to least significant and updates the internal state of the algorithm (Q_0 and Q_1) accordingly using point additions and doublings. Hence, at bit index i of k, the internal state depends on bits $\{0, \ldots, i\}$ of k.

Algorithm 6 Montgomery ladder

Input P , $k = (k_0, ..., k_{l-1})$
Output kP

1: $Q_0 \leftarrow O$
2: $Q_1 \leftarrow P$
3: **for** $i = 0$ to $l - 1$ **do**
4: $\quad Q_{1-k_i} \leftarrow Q_{1-k_i} + Q_{k_i}$
5: $\quad Q_{k_i} \leftarrow 2Q_{k_i}$
6: **return** Q_0

The general architecture of ECSM algorithms is layered as illustrated on Figure 3.1. The first layer relates to the point operations, namely addition and doubling on lines 4 and 5 in Algorithm 6. In our particular case-study, the points are represented using Jacobian coordinates and the corresponding point addition and doubling are described in Algorithms 7 and 8 respectively. In the point doubling algorithm, the multiplication by $a = -3$ is done using field subtractions, to save on one costly field multiplication. Regarding the second layer, point operations involve manipulating their coordinates, which belong to $\mathbb{F}_p$, using modular multiplications, additions and subtractions. These field operations

are in turn decomposed into 32-bit register operations that the device can carry out in one or a few clock cycles and correspond to the last layer of the ECSM architecture.

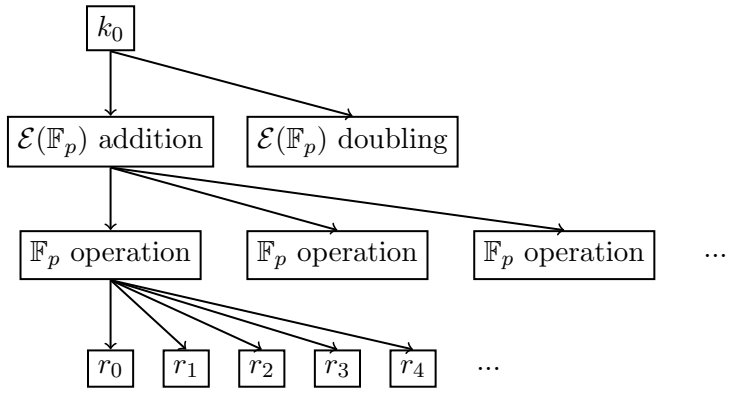


Figure 3.1: Layered view of a regular scalar multiplication (inspired by [134]).

Algorithm 7 Point addition using Jacobian coordinates

Input $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$
Output $P + Q = (X_3, Y_3, Z_3)$

$A \leftarrow Z_1^2$, $B \leftarrow Z_2^2$, $C \leftarrow X_1 B$, $D \leftarrow X_2 A$, $E \leftarrow C - D$, $F \leftarrow Y_1 B Z_2$
$G \leftarrow Y_2 A Z_1$, $H \leftarrow F - G$, $I \leftarrow E^2$, $J \leftarrow IE$, $K \leftarrow CI$
$X_3 \leftarrow H^2 + J - 2K$
$Y_3 = H(K - X_3) - FJ$
$Z_3 = Z_1 Z_2 E$
return (X_3, Y_3, Z_3)

Algorithm 8 Point doubling using Jacobian coordinates

Input $P = (X_1, Y_1, Z_1)$
Output $P + P = (X_2, Y_2, Z_2)$

$A \leftarrow X_1^2$, $B \leftarrow Y_1^2$, $C \leftarrow Z_1^2$, $D \leftarrow 3A + aC^2$, $E \leftarrow B^2$, $F \leftarrow 4X_1 B$
$X_2 \leftarrow D^2 - 2F$
$Y_2 \leftarrow D(F - X_2) - 8E$
$Z_2 = 2Y_1 Z_1$
return (X_2, Y_2, Z_2)

While the leakage of any and all field operations can be exploited by a side-channel attacker, the work of Poussier et al. focuses on field multiplications. In this respect, they implement field multiplication with

independent schoolbook Long Integer Multiplication (LIM) and modular reduction by the NIST P-256 prime. In particular, they target some of the intermediate results of single precision multiplications in the LIM. We provide in Figure 3.2 a simple example of a multiplication of two 64-bit inputs a and b using a 32-bit multiplier and ignoring single precision additions for simplicity. Concretely, a (resp. b) is decomposed into two 32-bit parts a_0 and a_1 (resp. b_0 and b_1) that can be stored in registers such that $a = a_0|a_1$ (resp. $b = b_0|b_1$), where $|$ denotes the concatenation. Computing $a \times b$ using a schoolbook method requires to compute all the cross products of 32-bit words using single precision multiplication available on the device. In this case, we first compute $a_0 \times b_0, a_0 \times b_1, a_1 \times b_0$ and $a_1 \times b_1$. However, each multiplication result of two 32-bit operands does not always fit on 32 bits, hence it is stored in two registers, one containing the 32 least significant bits, denoted as low in Figure 3.2, and one containing the 32 most significant remaining bits, denoted as high.

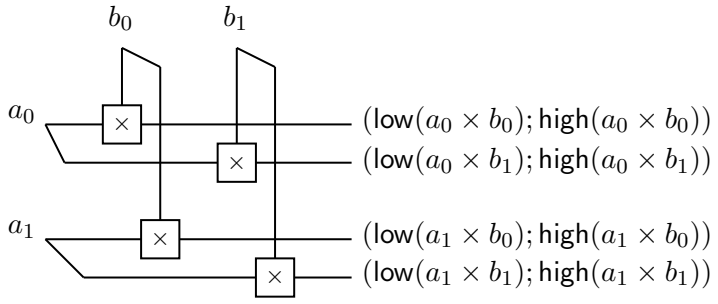


Figure 3.2: Simplified view of a long integer multiplication.

3.2 Horizontal differential analysis on ECSM

In this section, we describe the HDPA of Poussier et al. which exploits the leakage of the high parts of all single precision multiplication results, shown on Figure 3.2. As previously described, the internal state of the ECSM algorithm depends on the bits of k. After the first iteration, the values of Q_0 and Q_1 only depend on k_0. Based on this information, the attack divides the ECSM execution into a sequence of predictable values with two hypothesis based on the value of k_0. The values targeted are the high parts of all single precision multiplication results involved in field multiplications. The point addition and doubling operations require 25 field multiplications in total. Each field multiplication computes $8 \times 8 = 64$ cross products since 256-bit operands are stored in 8 32-bit words.

Note that while the point addition at line 4 of Algorithm 6 is always performed on the same operands Q_0 and Q_1, the operands' order also provides exploitable leakage since it affects the order in which the cross products are computed. This leads to $N = 25 \times 64 = 1600$ target registers. We detail below the tools used by Poussier et al. for each step of the attack following the measurement of the side-channel traces.

- **Leakage detection and mapping.** The points of interest corresponding to the target registers are identified using classic techniques such as the correlation-based test [56] assuming a simple HW model.
- **Leakage modeling.** Once the time locations of all targets are found, they can be modeled using classic Gaussian templates [34]. However, the number N of targeted intermediates requires the estimation of a large covariance matrix of dimension $N \times N$. For efficiency, the attack assumes the Independence of the Operations' Leakages (IOL) [77], which is a commonly used assumptions in the side-channel analysis of block ciphers. For our specific case-study, it was noted by Poussier et al. that fully characterizing the leakages' covariance matrix did not improve the attack's result. Additionally, modeling the leakage of 32-bit variables assuming an identity leakage model is too measurement intensive. As a solution Poussier et al. suggest to use a linear-regression-based approach using as a basis the 32 bits of the register values [145]. Concretely, this approach assumes that the deterministic part of the leakage of a register is a linear combination of its bits and that the noise is distributed according to a Gaussian distribution. In [134], the same profiling technique is used to model the individual leakages of all N registers. First, to find the basis $\mathbf{b}$, we require a vector $\mathbf{l}$ of n profiling samples of the leakage of the register r. Then, we compute the matrix $\mathbf{A} \in \mathbb{F}_2^{n \times 33}$, where the i-th line in $\mathbf{A}$ corresponds to the binary representation of the i-th sample value of r with its leakage being the i-th element of $\mathbf{l}$. All the binary representations in $\mathbf{A}$ are appended by 1. The linear basis vector $\mathbf{b}$ and the noise variance σ^2 are computed according to:

$$\mathbf{b} = (\mathbf{A}^{\mathrm{T}} \cdot \mathbf{A})^{-1} \cdot \mathbf{A}^{\mathrm{T}} \cdot \mathbf{l} \tag{3.1}$$

$$\sigma^2 = (\mathbf{l}' - \mathbf{A} \cdot \mathbf{b})^{\mathrm{T}} \cdot (\mathbf{l}' - \mathbf{A} \cdot \mathbf{b}) \tag{3.2}$$

  with $\mathbf{A}^{\mathrm{T}}$ the transpose of $\mathbf{A}$ and $\mathbf{l}' \neq \mathbf{l}$ another set of leakage samples for the values in $\mathbf{A}$ (or another corresponding matrix to $\mathbf{l}'$).

– **Information extraction.** The conditional probability of a leakage l_j knowing the value of its corresponding register r_j is evaluated as:

$$\Pr[l_j|r_j] = \mathcal{N}\left(l_j \middle| \sum_{i=0}^{31} r_{j,i} \cdot \mathbf{b}[i] + \mathbf{b}[32], \sigma^2\right) \tag{3.3}$$

where $r_{j,i}$ is the i-th bit of register r_j in the same order as $\mathbf{A}$.

– **Information processing.** Knowing the input point P, the bit value of k_0 is chosen as the one maximizing the product of the probabilities of the register leakages according to the following equation:

$$\Pr[k_0|(l_j)_{0\leq j<N}, P] = \prod_{j=0}^{N-1} \Pr[l_j \,|\, (r_j|k_0, P)] \tag{3.4}$$

Attacks exploiting the leakage of multiple intermediates against ECSM implementations are naturally performed by using an E&P strategy and recover the scalar bits in a recursive manner [14, 39]. To recover the bit at index i, the intermediate values are not only predicted based on the value of the bit at index i but also on the previously recovered bits and the same previous attack steps are repeated.

3.3 Problem statement and challenges

One recurrent problem of side-channel security evaluations is the large amount of different state-of-the-art attacks [165] which makes it prohibitive to try all of them: ECC implementations are no exception to this issue [30]. In this context, the HDPA described previously is an interesting one to investigate, since it belongs to a powerful type of attacks against ECC implementations. However, it comes at the cost of a complicated instantiation. First, it requires knowledge of the implementation under attack and the profiling of many operations of the ECSM, which is computationally intensive. In the following, we describe how this generic framework can be used for systematizing security evaluations and reducing their cost, by providing shortcut formulas to estimate the success rate of HDPA without performing it.

For this purpose, we draw our inspiration from the associated literature on symmetric cryptography. Indeed, shortcut formulas for success rate estimation in the case of block ciphers are already a deeply

investigated topic. In the simpler case of unprotected (more precisely, unmasked) implementations, efficient approximations of the success rate can typically rely on easy-to-compute metrics such as the Signal-to-Noise Ratio (SNR) [139, 59]. By contrast, for masked implementations, additional assumptions and/or metrics (e.g., the mutual information) are needed [109, 48, 53]. Since considering unprotected ECC implementations, we will next be in the former case and additionally exploit some of the ideas used in [77] for the analysis of multivariate/horizontal attacks. Precisely, we will adapt the SNR metric to the context of ECSM implementations and exploit it for the estimation of the success rate as a function of the number of targeted register leakages and the noise level.

3.3.1 Signal-to-noise ratio definitions

In general, the SNR of a device depends on the size of the register or bus (which defines the maximum signal), the adversary's guessing power (which defines the part that generates exploitable signal and the part that generates algorithmic noise) and the physical noise. It is defined as the variance of the (exploitable) signal divided by the noise variance. In this work, we consider a 32-bit device and therefore assume that all the bits of the bus can be predicted (so no algorithmic noise). In the context of a standard DPA where the full bus is targeted [113], this would lead to an SNR32 defined for a register indexed j as:

$$\mathsf{SNR32}_j = \frac{\underset{r_j \in \mathbb{F}_{2^{32}}}{\mathbb{V}} \delta_j(r_j)}{\sigma^2} \tag{3.5}$$

where δ_j is the deterministic (noise-free) part of the leakage function for a register r_j, $\mathbb{V}$ the variance operator and σ^2 the noise variance. Further assuming a HW leakage function for illustration, this leads to $\mathsf{SNR32} = \frac{8}{\sigma^2}$ (with $8{=}\frac{32}{4}$ the variance of a random 32-bit Hamming weight).

When considering a 32-bit implementation of the Montgomery ladder ECSM, the situation slightly differs from this standard DPA context. Indeed, in this case the register content typically depends on a single key bit (rather than 32 ones in the standard DPA case). Therefore, each target register can only take two values instead of every value of $\mathbb{F}_{2^{32}}$. A vertical DPA against an ECSM therefore boils down to distinguishing the leakage of two 32-bit values, whereas a HDPA tries to exploit multiple registers. Concretely, this means that certain registers lead to easier-to-distinguish leakages. Yet, in order to improve the efficiency of the security evaluations, we will also use an average metric to estimate the success rate (and track the distance between this estimate and the

success rate of concrete attacks). For this purpose, a first natural idea would be to consider a modified $\mathsf{SNR2}_j$ that captures the difference between the (noise-free) leakages of a register r_j for two scalar bit values and random elliptic curve points, which we define as:

$$\mathsf{SNR2}_j = \frac{\mathop{\mathbb{E}}_{P \in \mathcal{E}(\mathbb{F}_p)} (\delta_j(r_j|k_i = 0, P) - \delta_j(r_j|k_i = 1, P))^2}{\sigma^2} \tag{3.6}$$

Note that in order to predict the value of r_j during the processing of the i^{th} scalar bit, knowledge of the previous bits (index 0 to $i-1$) is required along with the input point P. This knowledge is equivalent to knowing the internal state of the ECSM, comprising of the points Q_0 and Q_1, after iteration $i-1$. For conciseness, in Equation 3.6 and in the following sections, we denote the variable $(r_j|k_0, \ldots, k_{i-1}, k_i, P)$ by $(r_j|k_i, P)$.

3.3.2 Preliminary observations and caveats

HDPA aims at exploiting the leakages of a large number of leaking registers for a single key bit. In the symmetric case, this can be viewed as targeting several leakage samples for a single key byte (e.g., both the input and the output of an Sbox). As a result, in this case we have that $\delta_i = \delta_j$ trivially implies $\mathsf{SNR32}_i = \mathsf{SNR32}_j$. This basically means that under the assumption $\delta_i = \delta_j$, the estimation of the $\mathsf{SNR32}$ is only required for a single register. Grosso and Standaert use this same assumption (of similar leakage functions for all their target intermediate computations) to speed up the computation of a multivariate mutual information to a univariate one [77].

Following this approach, a tempting strategy for ECSM evaluation would thus be to also assume that the leakage functions are similar for all the registers, leading to a constant $\mathsf{SNR2}$. To evaluate the soundness of this approach, Figure 3.3 shows the $\mathsf{SNR2}_j$ for each register r_j for $0 \leq j < 1600$, corresponding to the high 32-bit words of multiplication results as described in Section 3.2. The $\mathsf{SNR2}$ is evaluated for a HW leakage model, so exactly fulfilling the assumption of identical leakage functions, and averaged over 10,000 randomly sampled elliptic curve points. We observe that $\mathsf{SNR2}_j$ is not constant even when the leakage function is the same for all registers. These differences can be explained by the algebraic relations between the values computed when the scalar bit equals 0 and the values when this bit equals 1. For example, the regions of high $\mathsf{SNR2}$ on Figure 3.3 observed in the register index intervals [512,576] and [704,768] respectively correspond to the 9th and 12th field multiplication

in the point addition described by Algorithm 7. For a bit value, it computes E^2 and H^2 or $(-E)^2$ and $(-H)^2$ when the bit is flipped. This leads to bigger differences in the side-channel leakage, as the bits of the opposite of a field element modulus the NIST P-256 prime are almost all flipped. The peaks of zero SNR2 in register index interval [896,960] correspond to the 15th field multiplication in the point addition algorithm as well. It performs the operation Z_1Z_2 or Z_2Z_1 when the bit is flipped. Since the same elements are multiplied in both cases, 8 equal cross products appear during the computation of the LIM, thus leading to no information (SNR2 = 0).

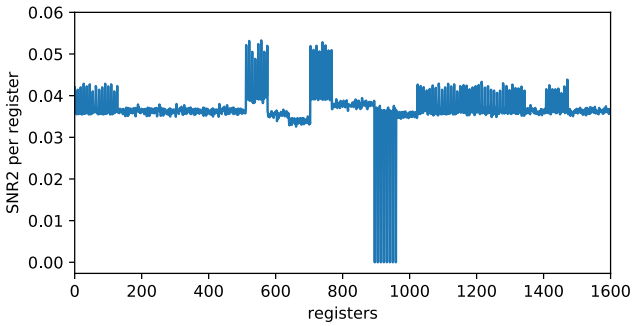


Figure 3.3: SNR2 of 1600 registers of field multiplications in the Montgomery ladder defined in Algorithm 6.

These observations imply that, as opposed to the symmetric case, a single register cannot be used to evaluate the security of ECSM implementations with respect to vertical DPA, even if the leakage function is the same for all the registers. This variation of the SNR2 highlights the fact that when performing these attacks, some leakage points are more interesting than others. So strictly speaking, such a simple evaluation is not possible for HDPA either.

3.3.3 The single trace attack scenario

Besides the previous caveat, another difficulty arises from the contradiction between HDPA that are essentially designed to succeed in a single-trace attack context (e.g., against a randomized key or an unknown scalar nonce) and the SNR2 metric which corresponds to an average amount of information collected over multiple points (and is therefore more in line with a vertical DPA).

In order to deal with this issue, we therefore start by defining the

register-specific amount of information that corresponds to the distance between a register value when the scalar bit equals 0 and its value when the scalar bit equals 1, for a fixed elliptic curve point. For register r_j and a point P, we denote this distance by $d_j(P)$, whose definition is given by Equation 3.7:

$$d_j(P) = (\delta_j(r_j|k_i = 0, P) - \delta_j(r_j|k_i = 1, P))^2 \tag{3.7}$$

Perfectly characterizing the security of an ECSM against HDPA naturally requires characterizing all the distances $d_j(P)$. Yet, and interestingly, we will next show that the two challenges described in this section (i.e., the fact that the SNR2 metric is register-dependent and that HDPA is primarily designed for single-trace attacks) can be mitigated concurrently. For this purpose, the main observation is that in view of the number of registers in ECSM implementations, it seems reasonable that the success of an attack targeting all the registers at once actually gets close to the average one. We next formalize this idea and describe the additional assumptions it requires.

3.4 Efficient success rate approximation

In this section, we show how to derive an approximation of the success rate with respect to the HDPA framework for one scalar bit recovery. Since we consider two equally likely Gaussian hypotheses, the SR when targeting a single register r_j for a given point P is computed as in [34] and given by Equation 3.8:

$$\mathsf{SR} = \Phi\left(\frac{\sqrt{(\delta_j(r_j|k_i = 0, P) - \delta_j(r_j|k_i = 1, P))^2}}{2\sigma}\right) \tag{3.8}$$

Next, we leverage the IOL assumption, which is used in the attack to improve its efficiency, and recall that it is a conservative one since deviations can only reduce the attack's effectiveness. It allows us to easily extend the previous formula to the case where an attacker would exploit N registers at once. Interestingly, it also re-enforces the analogy between vertical and horizontal DPA. Indeed, the IOL assumption divides the side-channel trace into N univariate samples, which roughly corresponds to a vertical DPA using N traces. As a result, similarly to the vertical DPA case [48], the SR of the horizontal DPA exploiting N samples, denoted by SR^N, is given by Equation 3.10:

$$\mathsf{SR}^N = \Phi\left(\frac{\sqrt{N \cdot \mathop{\mathbb{E}}_j \left(\delta_j(r_j|k_i = 0, P) - \delta_j(r_j|k_i = 1, P)\right)^2}}{2\sigma}\right) \tag{3.9}$$

$$= \Phi\left(\frac{\sqrt{N \cdot \mathop{\mathbb{E}}_j d_j(P)}}{2\sigma}\right) \tag{3.10}$$

As shown by Figure 3.3, the vertical signal $\mathsf{SNR2}$ is not constant across all registers as opposed to the symmetric case. This is also true for the horizontal signal $d_j(P)$, which will inevitably vary depending on the point and the targeted register. As a result, a strict approximation of the SR^N using Equation 3.10 would require to compute $d_j(P)$ for every single register. This requires a first step of leakage characterization [34, 145]. This step is quite intensive and the most time and data consuming in HDPA. Using Equation 3.10, the SR approximation is just as complex and tedious as performing the attack. This observation shows the need of additional assumptions in order to simplify the security evaluation.

3.4.1 Additional assumptions

Identical Leakage Functions assumption (ILF). We first assume that the leakage function is identical across all the target registers: $\delta_i = \delta_j = \delta$, for $i, j \in [0, N-1]$ since all resulting from the same assembly operation. Note that this is a common assumption that is also used in the security evaluation of masked implementations of block ciphers [77], side-channel reverse engineering [57] and the design of advanced side-channel leakage emulators [117].

Asymptotic Uniformity assumption (AU). We define the notion of *ideal distance* d_{id} as the squared difference between the noise-free leakages of two uniformly distributed values $V_1, V_2 \sim \mathcal{U}(\mathbb{F}_{2^{|r|}})$. More formally, given a leakage function δ, the ideal distance is given by Equation 3.11:

$$d_{\text{id}} = \mathop{\mathbb{E}}_{v_1, v_2} \left(\delta(v_1) - \delta(v_2)\right)^2. \tag{3.11}$$

Naturally, d_{id} can be seen as the *vertical* information provided by a register whose values are uniformly distributed when the input point varies. Our main assumption, the AU, states that the mean of the distances $d_j(P)$ over a large number of registers tends towards d_{id}, as given by Equation 3.12. Informally, it means that the average horizontal

information $d_j(P)$ for a fixed point P of a big enough number of registers is equal to the vertical information of a single uniformly distributed register:

$$\mathop{\mathbb{E}}_{j=0}^{N-1} d_j(P) \xrightarrow[N\to+\infty]{} d_{\mathrm{id}}. \tag{3.12}$$

3.4.2 Success rate shortcut formula

Using the two assumptions introduced in the previous subsection, we can efficiently estimate the SR of HDPA against an ECSM implementation. For the AU assumption, we further assume that the number N of exploited registers is big enough so that $\mathop{\mathbb{E}}_{j} d_j(P)$ is close to d_{id}. As a result, the SR^N approximation is boiled down to the computation of d_{id} and the estimation of the noise level σ. The success rate formula of Equation 3.10 is then trivially adapted to d_{id} as:

$$\mathsf{SR}^N = \Phi\left(\frac{\sqrt{N \cdot d_{\mathrm{id}}}}{2\sigma}\right). \tag{3.13}$$

Hamming weight leakage example. We illustrate this formula with an example based on a HW leakage function. The HW of a uniform random variable on $\mathbb{F}_{2^{|r|}}$ is approximately distributed as $\mathcal{N}(\frac{|r|}{2}, \frac{|r|}{4})$ with $|r|$ being the size of the considered register r. For $U_1, U_2 \sim \mathcal{U}(\mathbb{F}_{2^{|r|}})$, we have $\mathsf{HW}(U_1) - \mathsf{HW}(U_2) \sim \mathcal{N}(0, \frac{|r|}{2})$ and $(\mathsf{HW}(U_1) - \mathsf{HW}(U_2))^2 \sim \Gamma(\frac{1}{2}, |r|)$, where Γ denotes the Gamma distribution, here with shape parameter $\frac{1}{2}$ and scale parameter $|r|$. If the AU assumption holds, then the ideal distance d_{id} is equal to $\frac{|r|}{2}$. The corresponding success rate is given by Equation 3.14:

$$\mathsf{SR}^N = \Phi\left(\frac{\sqrt{N \cdot |r|}}{2\sqrt{2}\sigma}\right). \tag{3.14}$$

3.4.3 Potential invalidation of the assumptions

The previous equations express the SR of a HDPA as a function of its main parameters, which allows gaining intuition about how the complexity of HPDA scales. Yet, the concrete correctness of this proposal depends on the ILF and AU assumptions. In this section, we discuss how realistic these assumptions are, and identify issues that may contradict them in practice.

Algorithmic issue. Even if the leakage model is the same for all targeted registers, the SNR2 is not identical for all registers, as seen on Figure 3.3. The SNR2 and the distances $d_j(P)$ depend on the distribution of the intermediate values, the ECSM algorithm, the curve representation and the finite field arithmetic.

Physical issue. While commonly used in SCA, the ILF assumption is never fully verified in practice [65]. We might observe $\delta_i \neq \delta_j$ when $i \neq j$. This can introduce additional discrepancies among the registers' distances. It can impact the convergence of the mean distance to the ideal distance and thus the accuracy of the SR approximation using the AU assumption.

In the next sections, we show the validity of our approximations with respect to both issues. First, in Section 3.5, simulations are used to show that the AU assumption provides a valid approximation of the behavior of ECSM intermediate values. Next, in Section 3.6, we use real measurements to show that the physical errors are not too problematic for the accuracy of the SR approximation.

3.5 Simulated experiment: the algorithmic issue

In this section, we tackle the algorithmic issue, which relates to the distribution of the intermediate values of the ECSM and to what extent it deviates from a uniform distribution. For this purpose we use a perfect setting with a HW leakage function using simulated traces, to fulfill the ILF assumption so that conclusions are not affected by any physical aspect of a real device's leakage. We consider the reference implementation of the Montgomery ladder described in Section 3.2 and an attacker targeting all the $N = 1600$ multiplication results. For each execution with a random point P and a random scalar bit k_i, we are provided with the register values $(r_j)_{0 \leq j < 1600}$ along with their simulated leakages $l_j = \mathsf{HW}(r_j) + b_j$ where $b_j \sim \mathcal{N}(0, \sigma^2)$. The noise level is chosen to replicate the target device in Section 3.6 with $\sigma^2 = 440$.

3.5.1 Convergence to the ideal distance

On Figure 3.4, we plot the evolution of the average distances of 1000 random elliptic curve points. The y-axis corresponds to the mean distance computed over the registers indexed by the x-axis. Each colored curve

corresponds to one randomly chosen elliptic curve point. The horizontal black line represents the ideal distance $d_{\text{id}} = 16$. We can observe that even though we only consider 1600 registers over the large number of leaking registers of an ECSM execution, for different points P of the elliptic curve $\mathcal{E}(\mathbb{F}_p)$, the average distances $\frac{1}{N}\mathop{\mathbb{E}}_{j=1}^{N-1} d_j(P) = \frac{1}{N}\mathop{\mathbb{E}}_{j=0}^{N-1}(\delta(r_j|k_i = 0, P) - \delta(r_j|k_i = 1, P))^2$ tend towards the ideal distance d_{id} when N gets larger. This result shows that the AU assumption can roughly describe the behavior of the ECSM intermediate values' leakages, and the approximation is more and more accurate as the number of registers required for the success of the attack increases.

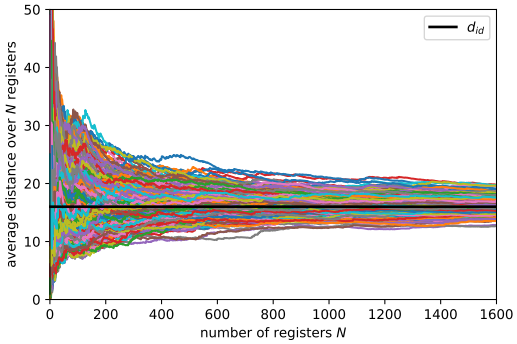


Figure 3.4: Convergence of the average HW distances to d_{id}.

3.5.2 Success rate approximation

The perfect simulated setting allows us to investigate the impact of the AU assumption on the SR approximation. Namely, the real success rate is only biased by the distances of register values and not by modeling errors. The results of the simulated experiments are illustrated in Figure 3.5. Each of the 20 colored curves corresponds to the SR of HDPA evaluated for one elliptic curve point (repeated 100 times), and the orange curve is the SR approximation using d_{id}. Knowing the noise variance σ^2, the latter is computed using Equation 3.14. The figure suggests that the approximation predicts well enough the real SR, showing that the algorithmic issue is not problematic for this particular implementation.

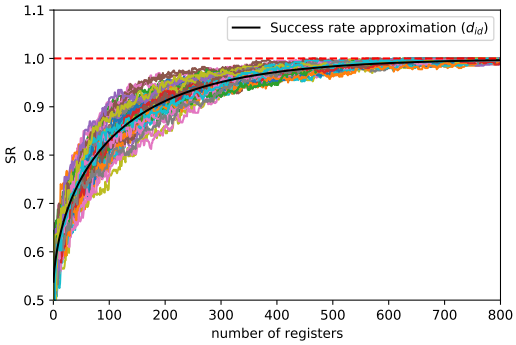


Figure 3.5: Comparison of the SR and its approximation on simulated HW leakages.

3.5.3 Impact of the noise

We performed the attack multiple times for different SNR32 values. First, the SNR32 of the target device in Section 3.6 (SNR32 = 0.0182), and additionally for SNR32 $\in \{0.1, 0.5, 1\}$. The results are depicted in Figure 3.6. The solid curves represent the SR of HDPA as function of the number of registers, and the dashed curves the corresponding approximations using the AU assumption. We draw attention to the gap between the real SR (solid line) and the SR approximation (dashed line) computed using Equation 3.14, which gets tighter as the SNR32 decreases. The bias introduced by the intermediate values of the ECSM makes the average distance over the small number of registers required for the success of HDPA for high SNR32 slightly deviate from the ideal distance. As the SNR32 decreases, this bias becomes smaller compared to the variance of the noise. Moreover, HDPA requires more registers to succeed, and thus requires to sum the distances over multiple registers which would tend towards the ideal distance as shown by Figure 3.4. This is an interesting result as we are mainly interested in the low SNR32 case, as it corresponds to high security devices that require this kind of worst-case analysis.

3.6 Real experiment: the physical issue

In this section, we investigate the impact of the ILF assumption on the accuracy of the SR approximation using the AU assumption. We use real measurements, where a different leakage function δ_j is expected for each

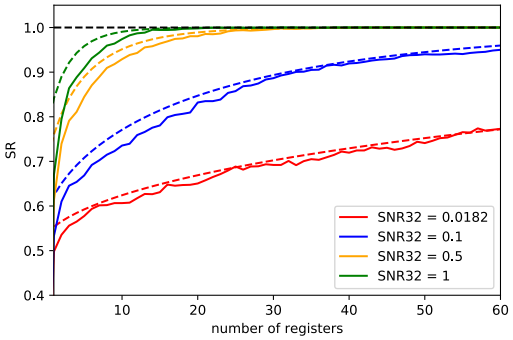


Figure 3.6: Impact of the side-channel noise on the SR approximation. Solid line for the attack's SR. Dashed line for the SR approximation.

leaking register r_j. Our experiments target the reference implementation similarly to the simulated case and device as in [134]. The target device is a 32-bit ARM Cortex-M4 micro-controller from the Atmel SAM4C-EK evaluation kit [2, 1] running at 100 MHz. The voltage variation was monitored using a 4.7 Ω resistor inserted in the power supply circuit of the chip. The trace acquisition was performed using a Lecroy WaveRunner HRO 66 ZI oscilloscope running at 200 MSamples/s per second. The executions of 10,000 scalar multiplications were recorded. For each of them, the measurement was triggered at the beginning of the execution and recorded the processing of one scalar bit. We performed HDPA on these traces such as described in Section 3.2 but assuming two different leakage models. First, a linear regression taking as a basis the HW of the leaking registers, similarly to the simulated experiment in the previous section. This yields for every register r_j, a leakage function of the form $\delta_j(r_j) = a_j + b_j \cdot \mathsf{HW}(r_j)$. Additionally, we performed HDPA for a linear regression based leakage model using a 32-bit basis, such as described in the original attack by Poussier et al. [134] and recalled in Section 3.2.

3.6.1 HW linear regression.

We study the influence of the physical issue on the convergence of the average distance across multiple registers towards the ideal distance. We start by evaluating the distance $d_j(P)$ for each register: $d_j(P) = (\mathsf{HW}(r_j|k_i = 0, P) - \mathsf{HW}(r_j|k_i = 1, P))^2$. We aim to compare the ideal distance to the mean distance for multiple different elliptic curve points. We evaluated the leakage model for the ideal distance using 20 random

register leakages. Figure 3.7 depicts the convergence of the average distance over the leaking registers for 1000 random elliptic curve points towards the ideal distance d_{id}. We observe that the distances indeed converge towards the ideal distance similarly to the simulated case in Figure 3.4 despite different leakage models for each individual register.

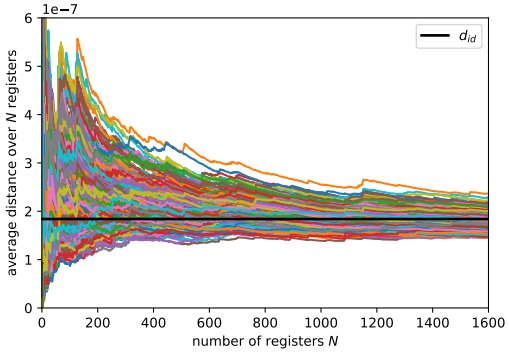


Figure 3.7: Convergence towards d_{id} of the average distances with different HW-based real leakage models for each register.

Additionally, Figure 3.8 shows the comparison between the real SR in blue of HDPA on real traces acquired from the target device previously described and its approximation in orange given by Equation 3.13. We averaged the SR over multiple points, so that conclusions are not affected by the algorithmic issue. We note that the approximation is still satisfactory but less accurate than in the simulated case. This is expected as the attack is performed on real side-channel measurements and the HW is not the most accurate leakage modeling strategy for this device, while the SR approximation assumes that the leakage model has been perfectly characterized.

3.6.2 32-bit linear regression.

We plot the convergence towards the ideal distance on Figure 3.9 for 1000 random elliptic curve points. We evaluated again the leakage model for the ideal distance using 20 random register leakages. We notice that despite having different leakage coefficients for each individual bit of the 1600 registers, the average distances still tend towards the ideal distance. This additional result further highlights the soundness of the AU assumption. Figure 3.10 shows the comparison between the SR of the HDPA in blue and its approximation in orange evaluated using

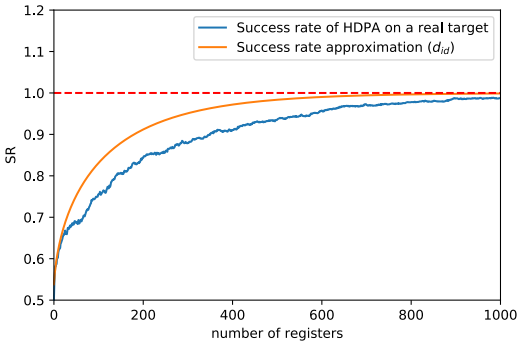


Figure 3.8: Comparison of the SR of HDPA and its approximation assuming a HW-based linear leakage model.

Equation 3.13. We notice that the SR approximation for a full basis linear regression attack is more accurate compared to the HW model case. This is due to the fact that the leakage of the considered device is best estimated by the second model.

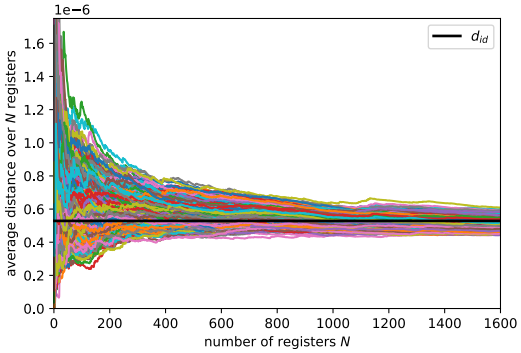


Figure 3.9: Convergence towards d_{id} of the average distances evaluated with a different linear regression function for each register.

3.7 Conclusion

Assessing the side-channel security of an implementation is a tedious task. This is particularly true for complex cryptosystems such as ECC for which numerous attack paths are possible. In this chapter, we de-

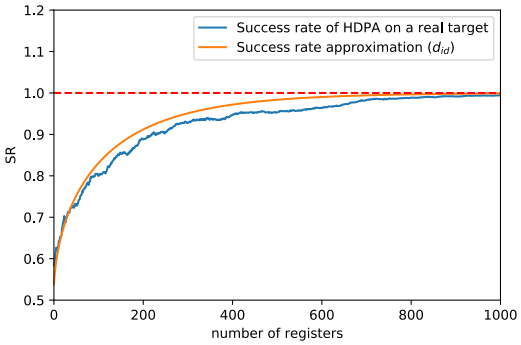


Figure 3.10: Comparison of the SR of HDPA and its approximation assuming a linear regression leakage model.

scribed a first methodology for analyzing the security of ECSM implementations against (close to) worst-case horizontal attacks. The proposed shortcut formula allows us to express the success rate of such attacks based on an easy-to-estimate (ideal distance) metric, in function the number of leakage samples exploited by the adversary (which depends on the register size, the field size and the number of field operations) and the noise level. This shortcut formula trades a bit of accuracy in the success rate estimation for considerable efficiency gains. As a future research direction, it could be extended to windowed algorithms that process multiple bits of the scalar during the same ECSM iteration.

Chapter 4

On the Worst-Case Security of ECC Point Randomization

In the recent years, the research on SCA has been shifting towards more powerful attacks with the goal to assess or approach the worst-case security level of cryptographic implementations. For example, the use of SASCA in the context of AES implementations [163, 76, 75] and more recently lattice-based cryptography [135], aim at exploiting more secret-dependent leakages that cannot be easily exploited by classic D&C attacks. Another example following this general direction is the attack of Poussier et al. [134], described in the previous chapter which is a nearly worst-case attack against ECSM implementations. Since this attack relies on the input point knowledge, the point randomization is a natural countermeasure against it, and its evaluation is an important research direction. Moreover, besides its relevance for side-channel protected ECC implementations, it is also of interest for pairing-based [96] and isogeny-based cryptography [104].

In this chapter, we assess the possibility of recovering the randomized input point so that any horizontal attack that relies on its knowledge can be applied. Our first contribution in this respect is detailed in Section 4.3, where we show how to efficiently apply SASCA in the case of point randomization by targeting the main field multiplications. For this purpose, we study the impact of different parameters of the implementation's factor graph. Then, in Section 4.4 we extend the local random probing model of Guo et al. [78] to our case study, in order to compare the efficiency of SASCA to a simpler and more efficient D&C method. Using this extension, we show that for realistic noise levels, SASCA does not provide a significant gain over the D&C strategy, particularly when the latter is augmented with enumeration. We then evaluate in Section 4.5 the required SNR for point randomization to be

secure against SCA. We show that it is relatively low, in comparison with the high noise levels required for practically-secure higher-order masked implementations [13, 32]. Additionally, in Section 4.6 we explore the concrete security of the point randomization countermeasure on an 8-bit device. In this respect, we consider two different options to implement long integer multiplication. Interestingly, we observe that the level of optimization of the multiplication significantly impacts the level of information leakage. In Section 4.7, we compare different elliptic curve coordinate systems with respect to the point randomization's side-channel resistance against advanced attackers. Finally, based on our detailed analysis of 8-bit implementations, we present in Section 4.8 guessing entropy bounds of the randomized point using the LRPM for hard to analyze 32-bit implementations.

4.1 Preliminaries and case-study

In this section, we provide some background about the point randomization countermeasure and field arithmetic and describe the working example that we study.

4.1.1 Point randomization

Randomizing the input point to the ECSM hinders horizontal attacks exploiting the leakage of the intermediates. We recall that, using homogeneous projective coordinates, any point $P = (x, y)$ on an elliptic curve $\mathcal{E}(\mathbb{F}_p)$ can be randomized by generating a random parameter $\lambda \in \mathbb{F}_p^*$ and replacing the representation of P by $(\lambda \cdot x, \lambda \cdot y, \lambda)$. This randomization is performed before every scalar multiplication takes place and increases the hypothesis space for each intermediate value from 2 possibilities to $2^{|\lambda|+1}$, with $|\lambda|$ the size of λ in bits. If $|\lambda|$ is large enough, it renders some attacks impractical. In the following, we restrict our analysis to homogeneous projective coordinates and later discuss in Section 4.7 other coordinate systems.

4.1.2 Field multiplication

The main operation performed during the point randomization procedure is a field multiplication of the random parameter λ and a coordinate of the input point. In practice, the input point to the ECSM is the elliptic curve public base point that we denote hereafter $G = (x, y)$. The implementation of optimized field multiplication is an important aspect

of elliptic curve systems. A field multiplication can be performed by independent long integer multiplication and modular reduction, i.e. first multiplying the $|p|$-bit operands and then reducing the $(2 \cdot |p|)$-bit result modulus the prime p. Alternatively, these steps can be interleaved to reduce the memory footprint. In practice, primes for elliptic curve applications are chosen to have a special form. They are called (generalized) Mersenne or Solinas primes and allow for very fast modular reduction as it can be performed by a series of a few additions and subtractions only [152].

Similarly to the previous chapter, we chose the NIST P256 curve as a practical example [130] and we consider a schoolbook LIM described in Algorithm 9, where n corresponds to the number of words or registers required to store a 256-bit variable. Accordingly, on an 8-bit (resp. 32-bit) architecture $n = 32$ (resp. $n = 8$). The register R_0 (resp. R_1) denotes the low (resp. high) part of the single precision multiplication result of a_i and b_j. The variable c denotes the carry coming from the previous addition. The modular reduction by the NIST 256 prime (a Solinas prime) is described in Algorithm 10. After the series of additions and subtractions, the result is contained in the range $[-4p, 5p]$. The final reduction is performed by adding or subtracting a small multiple of p. However, in practical implementations the result of a field operation does not have to be systematically and fully reduced modulus p. For efficiency, it can be simply be reduced modulus p to fit in the required number of registers to store a field element (e.g., 8 or 32 registers).

Algorithm 9 Schoolbook long integer multiplication

Input $a = (a_0, a_1, \ldots, a_{n-1})$ and $b = (b_0, b_1, \ldots, b_{n-1})$
Output $r = (r_0, r_1, \ldots, r_{2n-1})$ such that $r = a \times b$

1: **for** $i = 0$ to $2n$ **do**
2: $\quad r_i = 0$
3: **for** $i = 0$ to $n - 1$ **do**
4: $\quad$ **for** $j = 0$ to $n - 1$ **do**
5: $\quad\quad R_0, R_1 \leftarrow a_i \times b_j$
6: $\quad\quad r_{i+j} \leftarrow r_{i+j} + R_0$
7: $\quad\quad r_{i+j+1} \leftarrow r_{i+j+1} + R_1 + c$
8: $\quad\quad r_{i+j+2} \leftarrow r_{i+j+2} + c$
9: **return** $(r_0, r_1, \ldots, r_{2n-1})$

Algorithm 10 Modular reduction by the NIST P256 prime p

Input $s = (s_0, s_1, \ldots, s_{15})$ where s_i is a 32-bit word
Output $t = s \bmod p$

1: Define the 256 bit integers:
2: $t_1 = (s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)$
3: $t_2 = (0, 0, 0, s_{11}, s_{12}, s_{13}, s_{14}, s_{15})$
4: $t_3 = (0, 0, 0, s_{12}, s_{13}, s_{14}, s_{15}, 0)$
5: $t_4 = (s_8, s_9, s_{10}, 0, 0, 0, s_{14}, s_{15})$
6: $t_5 = (s_9, s_{10}, s_{11}, s_{13}, s_{14}, s_{15}, s_{13}, s_8)$
7: $t_6 = (s_{11}, s_{12}, s_{13}, 0, 0, 0, s_8, s_{10})$
8: $t_7 = (s_{12}, s_{13}, s_{14}, s_{15}, 0, 0, s_9, s_{11})$
9: $t_8 = (s_{13}, s_{14}, s_{15}, s_8, s_9, s_{10}, 0, s_{12})$
10: $t_9 = (s_{14}, s_{15}, 0, s_9, s_{10}, s_{11}, 0, s_{13})$
11: **return** $t_1 + 2t_2 + 2t_3 + t_4 + t_5 - t_6 - t_7 - t_8 - t_9 \bmod p$

4.2 Evaluation tools of the point randomization countermeasure

In this section, we describe two evaluation methods that can be used to assess the possibility for an attacker to recover the randomized point. In order to illustrate the two attack approaches, we first focus solely on the LIM involved in the field multiplication and later include the modular reduction. An abstract architecture of the schoolbook LIM of 256-bit operands on an 8-bit device is given in Figure 4.1. The variables $(\lambda_i)_{0 \leq i < 32}$ represent the words of the secret parameter λ that the attacker aims to recover, $(x_i)_{0 \leq i < 32}$ to the words of the x-coordinate of the public base point and $(r_i)_{0 \leq i < 64}$ to the words of the result. Following the previous section, R_0 and R_1 are respectively the low and high registers of single precision multiplication results, and since every word of λ is multiplied once by every word of x, 2048 such registers are computed. They are then combined in the accumulation block in Figure 4.1 corresponding to the additions on lines 6-8 in Algorithm 9 to yield the 64-word result r.

Given this abstract view of a multiplication, we identified two profiled single-trace side-channel attacks that can be used to defeat the point randomization countermeasure. First, a D&C attack that targets words of the secret independently (similar to how classic side-channel attacks target block cipher subkeys or Sboxes) and a SASCA [163], i.e. a BP-based attack that can potentially exploit all computations depending on the secret. The parts of the multiplication algorithm that each attack exploits are shown on Figure 4.1. Our aim is to compare them

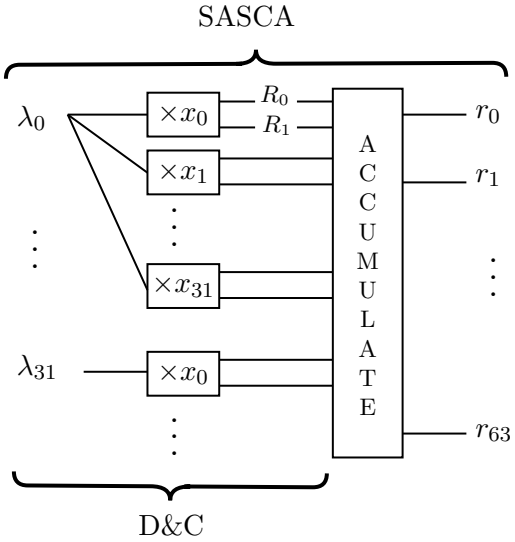


Figure 4.1: Abstract architecture of a schoolbook long integer multiplication.

to assess the most optimal security evaluation strategy of the point randomization countermeasure. We emphasize that both methods can be adapted to any point randomization procedure and field multiplication algorithm.

4.2.1 Horizontal D&C attack with enumeration

The first D&C approach is efficient since it targets each word of λ individually and only exploits the leakage of the intermediates that depend on a single word. The same attack process is repeated for all words, then an enumeration algorithm such as described in [161] or in [133] is used to list the most probable values of λ. Following, the description of the multiplication algorithm on Figure 4.1, a word λ_i is multiplied by the 32 known words of the x-coordinate of the base point $(x_j)_{0\leq j<32}$. By making a hypothesis on λ_i an attacker can predict the 64 corresponding values of R_0 and R_1. He then exploits the leakage of these intermediates in addition to the direct leakage on λ_i. The side-channel information is extracted by characterizing the joint conditional distribution:

$$\Pr[\lambda_i|\mathsf{L}(\lambda_i), \mathsf{L}((\lambda_i \times x_0)\%2^8), \ldots, \mathsf{L}((\lambda_i \times x_{31})/2^8)] \tag{4.1}$$

where $\mathsf{L}(v)$ denotes the side-channel leakage of a value v, % the modulus operator and / the integer division. Additionally, to simplify the leakage modeling, we can leverage the IOL assumption [77] which assumes the

independence of the targeted intermediates and the independence of the noise. The independence of the intermediates' distributions was verified for our case-study by inspecting their covariance matrix. As a result, the joint distribution described by Equation 4.1 can be factorized into:

$$\Pr[\lambda_i|\mathsf{L}(\lambda_i)] \times \prod_{j=0}^{31} \Pr[\lambda_i|\mathsf{L}((\lambda_i \times x_j)\%2^8] \times \Pr[\lambda_i|\mathsf{L}((\lambda_i \times x_j)/2^8)] \quad (4.2)$$

4.2.2 Soft analytical side-channel attack

A SASCA can exploit a larger number of intermediates in comparison to the D&C attack previously described. This is illustrated on Figure 4.1, where we highlight that SASCA can utilize the leakage coming from the additions in the accumulation block following the single precision multiplications. The first step of this approach is to build the factor graph of the multiplication as explained in Section 2.5. The involved factors are shown on Figure 4.2 and are: multiplications $f_\times$, additions f_{add} and additions taking into account the previous carry f_{adc}. They are defined as follows:

$$f_\times(x, y, R_0, R_1) = \begin{cases} 1 \; if \; R_0 = (x \times y)\%2^8 \; and \; R_1 = (x \times y)/2^8, \\ 0 \; otherwise. \end{cases}$$

$$f_{\mathsf{add}}(x, y, r, c) = \begin{cases} 1 \; if \; r = (x + y)\%2^8 \; and \; c = (x + y)/2^8, \\ 0 \; otherwise. \end{cases}$$

$$f_{\mathsf{adc}}(x, y, c, r, c') = \begin{cases} 1 \; if \; r = (x + y + c)\%2^8 \; and \; c' = (x + y + c)/2^8, \\ 0 \; otherwise. \end{cases}$$

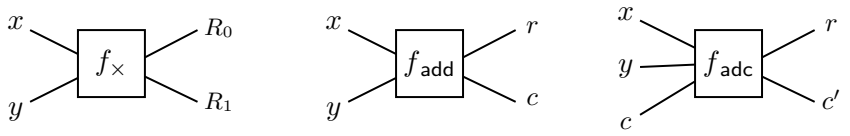


Figure 4.2: Factors involved in the long integer multiplication.

The next step of the attack is to apply the belief propagation algorithm to extract information from the intermediates that depend on multiple words of λ and combine it efficiently. The optimized information combination approach of SASCA implies that it is an appropriate tool to approximate the worst-case security of cryptographic implementations.

4.3 Analysis of the field multiplication factor graph

Prior to comparing the attacks identified in Section 4.2, we study the application of SASCA to a LIM factor graph. The nodes included in the graph and its structure can impact the performance of the BP algorithm. In the following, we investigate the impact of these characteristics. For this purpose, we build the factor graph of the first block from the pseudo-assembly description of the operand-caching multiplication of Hutter and Wenger on an 8-bit micro-controller [87, Appendix A]. The operand-caching multiplication is an optimized schoolbook multiplication that minimizes the number of operand word loads. It is specifically designed for small embedded devices in order to improve efficiency by minimizing memory operations. This graph that we denote as G is presented on Figure 4.3, where a factor $f_{\times}^{x_j}$ denotes a multiplication factor $f_{\times}$ with a constant and known value x_j. While factor graphs are undirected, we use directed edges to explicit inputs to and outputs from the corresponding operation to a factor. Given the size of G, it is possible to run multiple experiments of the BP algorithm to get meaningful averaged results. The conclusions drawn from our analysis can be generalized to the full graph and any multiplication that follows the abstract architecture on Figure 4.1 because of its very regular structure.

There are two aspects to consider before running BP on this graph that we detail and investigate hereafter. First, G is a cyclic graph and BP convergence to the correct marginals is not guaranteed [110]. Second, while for previous applications of SASCA to AES [163, 76, 75] and lattice-based encryption [135], the factor graphs contained factors of at most degree 3, G contains factors of degree 4 and 5 due to the additions with carry propagation. Although carry bits are small variables, we note that their values and possibly errors on their values may ripple through all the following steps of the computation.

To investigate the effect of cycles and carry bits, we construct four other factor graphs. $\mathsf{G\_no\_cycles}$ is an acyclic version of G built by following the strategy from Green, Roy and Oswald [75]: removing factors causing the cycles and severing any orphaned edge or node. $\mathsf{G\_no\_carry}$ is constructed by deleting carry bit variable nodes and integrating both possible values of a carry into the factors as described by the diagram on Figure 4.4, where the c (resp. c') variable node corresponds to the input (resp. output) carry. The new factor $f_{\mathsf{adc\_noc}}$ takes into account the two possible values of the input carry c and is defined as:

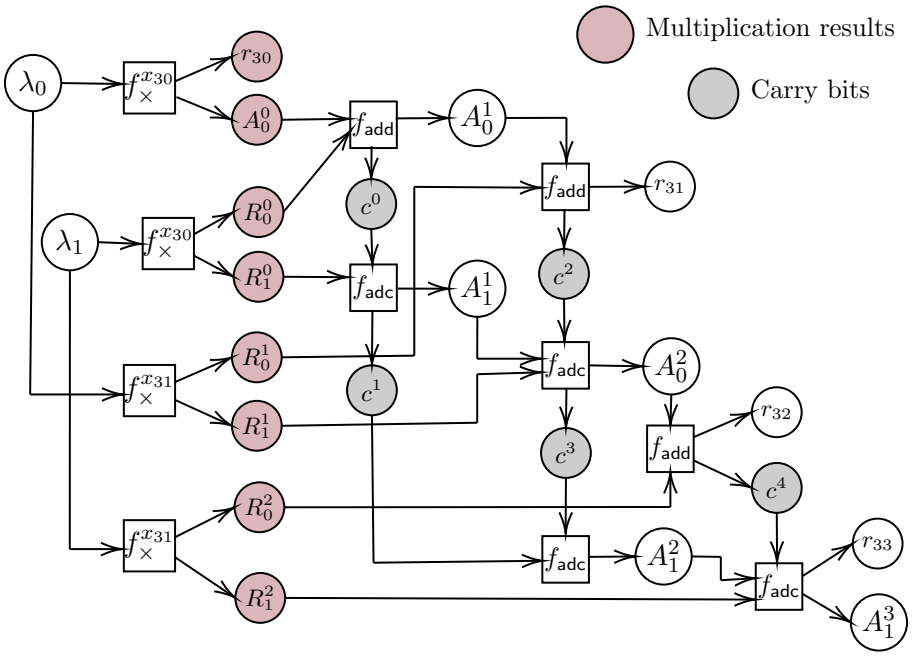


Figure 4.3: Factor graph G of the first part of the schoolbook operand-caching multiplication of Hutter and Wenger [87, Appendix A].

$$f_{\mathsf{adc\_noc}}(x,y,r) = \begin{cases} 1 \; if \; r = (x+y)\%2^8 \; or \; r = (x+y+1)\%2^8, \\ 0 \; otherwise. \end{cases}$$

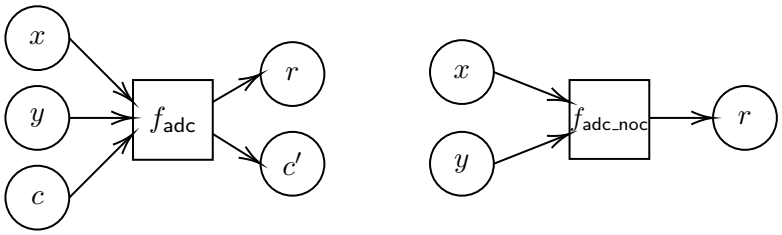


Figure 4.4: Conversion of addition with carry factor f_{adc} to factor $f_{\mathsf{adc\_noc}}$.

This transformation is given by the diagram for the addition with carry operation but for the addition factor f_{add}, the ouput carry is simply ignored. Finally, G_no_carry_no_cycles is G_no_carry where remaining cycles have been removed.

To compare the different graphs, we use simulated HW leakages for all intermediate variables with noise distributed according to $\mathcal{N}(0, \sigma^2)$. We run belief propagation on the four graphs and estimate the single trace attack SR (average across λ_0 and λ_1) for different noise levels. The results of these experiments are plotted on Figure 4.5. First, we notice that for high SNR values, the cyclic graphs (G and G_no_carry) perform better than the acylic ones (G_no_cycles and G_no_carry_no_cycles). This can be explained by the fact that cycles typically exacerbate side-channel errors but in this case are less detrimental since errors from side-channel observations are less likely to occur for high SNR. Additionally, the acyclic graphs are constructed by removing factors which contribute the most to the connectedness of the graph which generate the cycles and as a result lose some information. When moving to the (more realistic) low SNR range (below 2), G_no_carry yields marginally better results, as it still benefits from the additional information provided by factors in cycles, and is also not prone to errors on carry bits. For even lower SNR (below 0.05), the experiments indicate that the best graph option is G_no_carry_no_cycles, which reflects previous observations on the impact of cycles and carry bits.

4.4 Comparison of SASCA and D&C attacks

In this section, we begin our investigation of the security level that can be achieved by the point randomization countermeasure by comparing

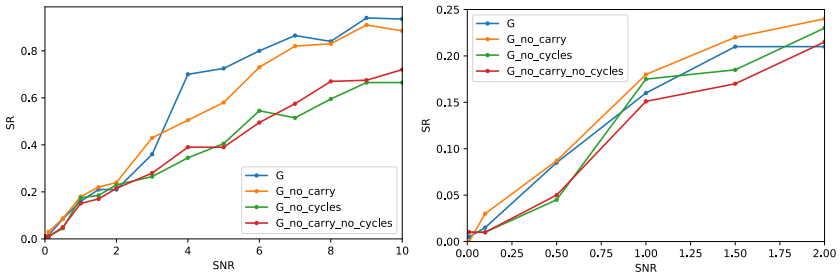


Figure 4.5: Average single trace SR as a function of the SNR for different factor graphs. Right: close-up for low SNR values.

SASCA and a D&C attack applied to a field multiplication, both described in Section 4.2. Similarly to the previous section, we still consider the 8-bit implementation of the operand caching multiplication. Since SASCA comes at a higher computational cost than a D&C attack, it is worth evaluating its advantage over simpler and more efficient strategies. The LRPM described in Section 2.5 is a heuristic shortcut tool to upper bound the information that can be extracted from running the BP algorithm on a factor graph [78]. For the purpose of the comparison, we make use of the LRPM's efficiency to process the sizable factor graph of the full field multiplication and introduce new extensions to the LRPM information propagation rules such that it is applicable to the target factor graph.

4.4.1 LRPM on field multiplications

To investigate the relevance of the LRPM, Guo et al. [78] consider the AES as a target, for which all atomic operations have one output only and LRPM rules apply straightforwardly. To apply the LRPM to the full factor graph of a field multiplication, we extend the information propagation rules to operations with multiple outputs. This new rule is described below and is based on the factorization principle of BP. An example factorization for an addition factor is given in Figure 4.6. The f_{add} factor of degree 4 can be factorized into two smaller factors of degree 3, f^1_{add} and f^2_{add} which are defined as:

$$f^1_{\mathsf{add}}(x,y,r) = \begin{cases} 1 \; if \; r = (x+y)\%2^8, \\ 0 \; otherwise \end{cases}$$

$$f^2_{\mathsf{add}}(x,y,c) = \begin{cases} 1 \; if \; c = (x+y)/2^8, \\ 0 \; otherwise \end{cases}$$

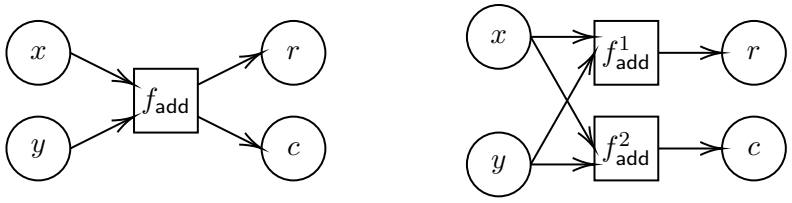


Figure 4.6: Factorization of an addition factor.

The new LPRM rules for information propagation for the addition factors (or factors with multiple outputs) are deduced from this factorization into one output factors. For the outputs r and c, the rules remain unchanged and their information coming from the f_{add} factor is simply the multiplication of the information on x and y. However, the information propagation rules towards the inputs is different. We consider x in the following, but the same rules apply by symmetry to y. Looking at the right side of Figure 4.6, we can apply the standard LRPM rules for one output factors to compute:

$$\mathsf{MI}_{f^1_{\mathsf{add}} \to x} = \mathsf{MI}_y \times \mathsf{MI}_r \text{ and } \mathsf{MI}_{f^2_{\mathsf{add}} \to x} = \mathsf{MI}_y \times \mathsf{MI}_c$$

Since information at variable node is summed we have:

$$\mathsf{MI}_{f_{\mathsf{add}} \to x} = \mathsf{MI}_{f^1_{\mathsf{add}} \to x} + \mathsf{MI}_{f^2_{\mathsf{add}} \to x} = \mathsf{MI}_y \times (\mathsf{MI}_r + \mathsf{MI}_c)$$

The LPRM estimates an upper bound on the information that can be extracted on a variable, and to avoid too pessimistic upper bounds, Guo et al. introduce the notion of loss coefficient [78] (Guo et al. use the term loss factor, we use loss coefficient to avoid confusion with the factors in the graph). A loss coefficient $\alpha \in]0, 1[$ is defined as a constant value that is multiplied by the information values propagated in the graph to yield closer values to the real information that would be observed in practice. For instance, Guo et al. consider $\alpha = 0.9$ as a loss coefficient for XOR factors and argue that this coefficient is independent of the noise level.

Regarding our case-study, we estimated experimentally the loss coefficients for every kind of operation in the graph, by assuming a HW leakage function with very low noise ($\mathsf{SNR} = 10$), as the ratio between the information computed after performing BP and the one predicted by the LPRM rules. We aim to capture a realistic upper bound so we simply consider the rounded maximum value observed for the loss coefficient over our simulations. The use of loss coefficients is also motivated by the addition with carry operations present in the factor graph we analyze. In particular, we notice that some information is propagated from 8-bit variables to carry variables that can only encompass one bit

of information, thus the LRPM is likely to overestimate the information propagated towards the carry bits.

Prior to using the LRPM to compare SASCA and D&C attacks, we confirm on Figure 4.7 that the LRPM's MI predictions fit the experimental SR on the factor graph G and its variants. We ran the LRPM assuming MI values that correspond to the simulated SR experiments and we focus on reasonable and realistic noise levels for software implementations. The left part of Figure 4.7 corresponds to the MI on one word of λ of each graph as a function of the SNR, and the right part to the experimental SR. The efficiency orderings of the different graphs in terms of MI and SR are similar. Slight discrepancies might be due to the experimental estimation of the SR. Decisively, the LRPM and the SR estimations agree on the fact that the best graph option is G_no_carry for a reasonable SNR < 2, while for even lower SNR all seem to perform similarly.

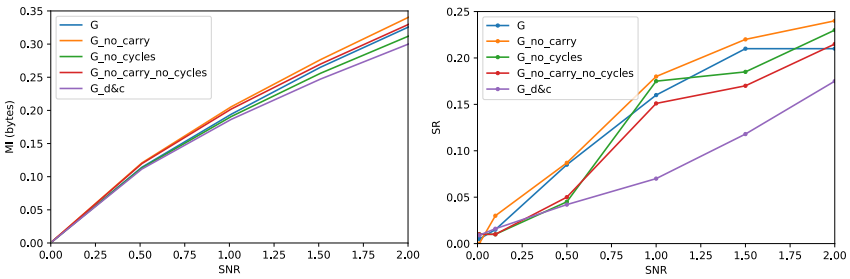


Figure 4.7: Left: MI evaluated with the LRPM with loss coefficients for the different graphs. Right: the SR of SASCA for different graphs.

4.4.2 SASCA vs. D&C

Based on our previous results, we build the full graph of the LIM with carry bits integrated into factors. Notably, this graph option is not only the best based on the previously shown experiments and LRPM predictions, but also the most pragmatic one. Indeed, the full graph of the LIM is highly connected. Attempting to remove all cycles will render the graph very close to the one exploited by the D&C strategy, with only the multiplication factors remaining. Accordingly, we build the full graph which consists of 1024 multiplication factors and 3216 addition factors, then the factor graph corresponding to the D&C attack that only contains the multiplication factors and related variables. We addi-

tionally build the full graph of the randomization procedure including the modular reduction implemented as suggested in [130] and described in Algorithm 10. In comparison to the efficient D&C attack, running the BP algorithm for a single iteration on the factor graph of the LIM would naively require more than 2^{37} operations, or more than 2^{29} operations with specific factor message passing optimizations.

For all three graphs, we use the LRPM to upper bound the information extracted in bytes on a word of λ. Results are given in Figure 4.8. We observe that both BP and D&C attacks reach the maximal information of one byte across all words of λ when the SNR exceeds 0.4. The D&C attack succeeds right after SASCA and, for lower SNR values, the gain is negligible considering the running time and the effort required to mount a BP-based attack. This is in-line with the results from [78] (e.g., for the AES): when the number of attack traces is too low (e.g., in a DPA), SASCA does not provide any gain over a D&C attack. Things naturally get worse for lower SNR values.

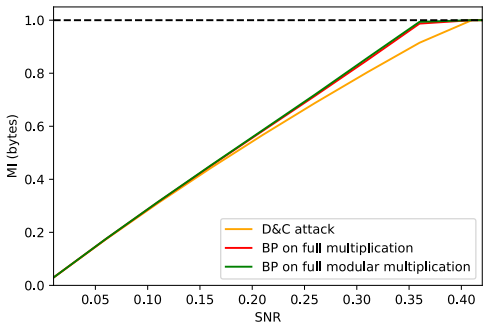


Figure 4.8: Comparison of SASCA on the full graph of the field multiplication, including the modular reduction, and the D&C attack on register multiplications.

Our results therefore indicate that classical D&C attacks (on our particular target) come very close to the worst-case attack for low SNR cases (which are the most interesting ones for side-channel investigations). This fact can be additionally highlighted by the limited information provided by the addition of the modular reduction to the factor graph compared to solely the multiplication. An analogy can be made with the AES MixColumns: since the operations contained in these examples (MixColumns, modular reduction by a Mersenne prime) diffuse already limited information (by combining noisy leakages of multiple intermediates), they do not contribute significantly to the overall infor-

mation on the targeted secret for low and reasonable SNR values.

Overall, this result shows that SASCA, which exploits all the available information, barely improves the results compared to a D&C attack. Moreover, as SASCA does not allow optimal enumeration [76] (since the words of λ are extracted jointly and all enumeration strategies assume the independence of the parts of the secret), adding the use of computational power to D&C attacks mitigates even more the already small gap between them. As a result, the rest of this chapter considers the D&C strategy in order to evaluate the security of the point randomization countermeasure. This naturally raises the question of how much noise is necessary to make the attack fail, which we investigate in the following section.

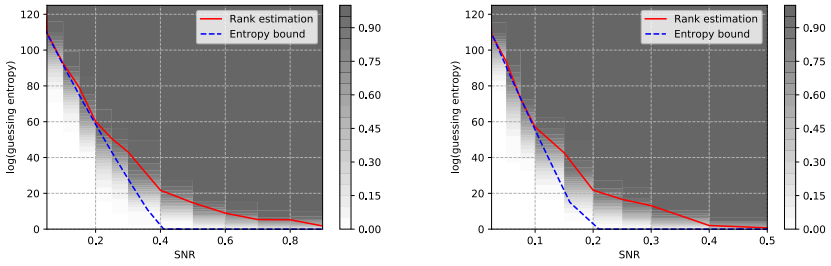
4.5 Security graphs and necessary noise levels

In this section, we investigate the nearly worst-case security of the point randomization countermeasure. For this purpose, we use the D&C strategy, identified in the previous section to achieve comparable efficiency to the worst-case BP-based attack. Since the D&C attack strategy is extremely efficient computationally (in comparison to SASCA), it allows investigating different implementation cases. For our experiments, we chose a homogeneous projective coordinates system to represent the points and later discuss the case of Jacobian projective coordinates. A homogeneous coordinates system allows for a very efficient point randomization procedure as it requires at most two field multiplications. We consider the case where both the affine coordinates of a point are randomized, and additionally the fast parallel point addition and doubling Montgomery ladder from Fischer et al. [60], where only the x-coordinate is required and thus randomized. The field multiplication is as previously described: a long integer multiplication followed by a modular reduction. We also take into account a 256-bit randomizing parameter λ and a 128-bit alternative. For both cases, we take as before the curve parameters of the NIST P256 for a concrete example. We recall that the choice of the curve is arbitrary and does not impact the attack strategy.

The goal of this section is to provide a characterization of the security level expected as function of the measurements' SNR. For this purpose, we plot security graphs based on the work of Veyrat-Charvillon, Gérard and Standaert [162] for the different cases of the point randomization procedure. These graphs are produced by performing 100 independent attacks and rank estimations for different SNR values. We use

the histogram-based method of Glowacz et al. to compute the ranks [68]. Using the sampled ranks provided by these experiments, the rank distributions can be easily estimated using kernel density estimation. The most interesting conclusions can be deduced from the Cumulative Distribution Function (CDF) of the rank. The CDF of the rank for a specific SNR tells us about the probability that the rank of the secret lies above a certain enumeration effort, and thus the probability for an adversary to recover the secret. This is visually represented by a gray-scale on a security graph: the darker (resp. lighter) the zone is, the higher (resp. lower) is the probability of recovering the secret.

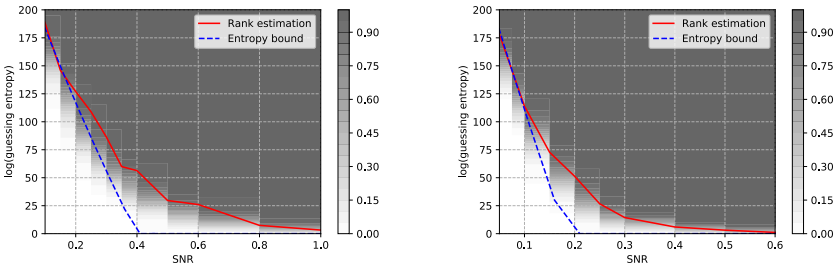
For the following results, we performed D&C attacks and corresponding rank estimations on simulated leakages of every 8-bit word of λ and every result of a register multiplication following the standard side-channel model: HW leakage and Gaussian noise. This leads to 65 leakages per target byte (one for the byte itself and 64 for the 2 parts of the 32 multiplication results by the 32 known bytes of the x-coordinate of the base point). We produce on Figure 4.9 the security graph for a 128-bit λ, when randomizing only the x-coordinate of the base point on Figure 4.9(a) and when randomizing both coordinates on Figure 4.9(b). Figure 4.10 gives the corresponding security graphs for a 256-bit λ.



(a) x (b) x and y.

Figure 4.9: Security graph of the point randomization countermeasure for $\lambda \in \mathbb{F}_{2^{128}}$

The red line corresponds to the $\log_2$ of the GE. The security graphs shown in Figures 4.9 and 4.10 encompass a great deal of information on the security of the target against D&C attacks. We give an example of a conclusion that can be deduced from one of these security graphs: to achieve security against an adversary attacking only the x-coordinate randomization using a 256-bit parameter, who can enumerate up to 2^{50} candidates, the SNR needs to be lower than 0.3 based on the results



(a) x (b) x and y.

Figure 4.10: Security graph of the point randomization countermeasure for $\lambda \in \mathbb{F}_{2^{256}}$

displayed on Figure 4.10(a). The comparison of Figure 4.9(a) and Figure 4.9(b) highlights the expected fact that the information is simply doubled when exploiting $\lambda \times x$ and $\lambda \times y$ compared to only $\lambda \times x$. Subsequently, to get the same level of security, the SNR has to be halved. The same applies to Figure 4.10(a) and Figure 4.10(b). Moreover, the security graphs for the 128-bit case and the 256-bit case illustrate the trade-off between the randomness requirements (the size of λ) and the side-channel noise necessary to make the point randomization robust against nearly worst-case adversaries.

Additionally, since the LRPM provides an upper bound on the information that can be extracted from a factor graph (in this case of the D&C attack), in all security graphs we plot the corresponding LRPM estimated remaining entropy in dashed blue line as a lower bound of the actual GE. While the relation between the entropy H and the GE is not defined, we showed in [10] that H can be used to heuristically predict the GE. In addition, this bound is quite helpful particularly when performing multiple attacks and rank estimations is not possible. It is tighter for low SNR, since the LRPM approximations are more accurate for high side-channel noise.

4.6 Experimental evaluations

In this section, we study the real world security of the point randomization on a practical embedded device. For this purpose, we analyze two different implementations of a long integer multiplication both written in assembly. First, a naively implemented schoolbook multiplication with two nested unrolled loops and no specific optimizations. The sec-

ond multiplication is the operand-caching multiplication of Hutter and Wenger [87] that we presented previously.

4.6.1 Device, measurement setup and attack description

We performed our experiments on an 8-bit AVR ATmega328p microcontroller mounted on an Arduino UNO board running at 16 MHz. The number of clock cycles required to perform the modular multiplication with the schoolbook and the operand-caching methods are respectively around 25k and 15k. Using a custom probe, the measurements are captured on a PicoScope5244D oscilloscope synchronized with a trigger at the beginning of each execution, at a sampling rate of 125 MSamples/s. The traces are preprocessed using amplitude demodulation.

Hereafter we describe the D&C attack applied to both implementations. Since leakage of a specific byte does not span the whole trace but is only located in very specific regions, we first performed a preliminary POI selection step. The goal of this step is to find the samples that correspond to leakages of intermediate values depending on the target byte. These leakages are found by applying the correlation-based leakage detection test [56] with a threshold set to 5. While we only compute the correlation coefficient between the leakages and every word of λ or the high and low parts of multiplication results, some additional variables involved in the accumulation block of the LIM might be picked up by the correlation test. This is due to dependencies between the intermediate values and provides additional information on λ. Taking into account the previously selected POI, we then use a dimensionality reduction technique, namely Principal Component Analysis (PCA) [6], to further reduce the number of dimensions. This can be done since as the coordinate of the base point is fixed and known, the leakage only depends on the bytes of λ. Using the compressed traces, we build multivariate Gaussian templates using 49k traces based on the values of the bytes. Note that while the noise is independent for the attack on simulations presented in Section 4.5, it is not perfectly the case for real traces. We thus take the dependency into account in order to improve the results and perform a single-trace Template attack on each secret byte. Finally, we combine the results of all the bytes using the Glowacz et al. histogram-based rank estimation algorithm [68].

4.6.2 Classic schoolbook multiplication

In this section, we show the results of the D&C attack on the non-optimized schoolbook multiplication. This multiplication loads every byte of the secret 32 times, displaying notable leakages on λ.

As a first step, we examine the bytes' guessing entropies in function of the number of PCA components retained for the compression of the traces. These results are shown in Figure 4.11. Typically, when applying PCA to side-channel leakages of symmetric cryptography implementations, the relevant information is located in a few principal directions (e.g., the recent work in [28] uses 10). For our specific target, more dimensions are useful, as shown by Figure 4.11. This is presumably due to the large amount of intermediate values that relate to the secret bytes, and the amount of different single-precision operations manipulating these intermediates. Additionally, this might also be due to the fact that each single-precision multiplication result is independent from all other multiplications, as discussed in Section 4.2. From Figure 4.11, we observe that the logarithms of the guessing entropies of all bytes decrease similarly and are mostly below 4 bits (less than 16 candidates) when the number of components is above 350. The guessing entropies keep on decreasing as the number of components increases.

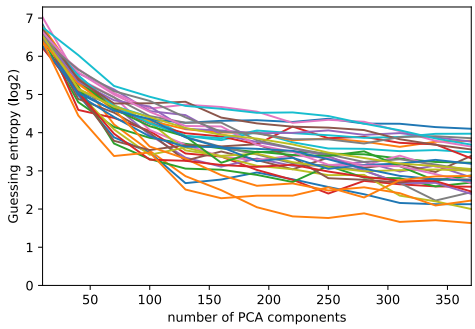
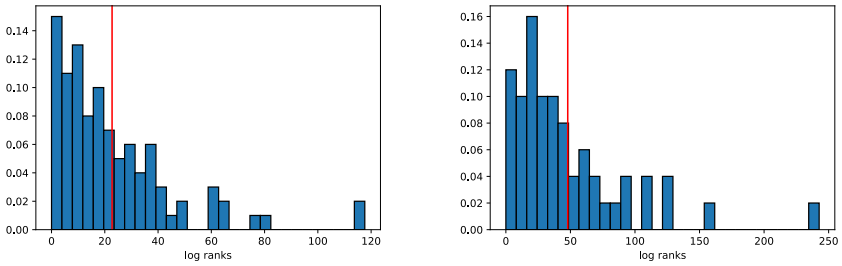


Figure 4.11: Logarithm of the guessing entropy of the 32 bytes of λ as function of the number of PCA components for the non-optimized schoolbook multiplication.

The next step of the evaluation is to assess if a single-trace recovery is feasible on the full value of λ. For each execution, we combine the results obtained from the independent attacks on the bytes of λ and plot the results of rank estimation on Figure 4.12. We show the distribution of the logarithms of the ranks. The red vertical line denotes the mean log rank observed. We also focus on two sizes for λ, namely 128 and 256 bits, in order to evaluate the impact of the D&C attack on the required randomness for each execution. First, based on Figure 4.12(a), for a 128-bit randomizing parameter, we observe that approximately half of the randomized points can be recovered with a minimal enumeration of

less than 2^{16}. The results for a 256-bit randomizing parameter are given on Figure 4.12(b). In this case, while the rank of the secret is higher than for the 128-bit case, the implementation can still be considered vulnerable with an average rank of 50 bits. Moreover, some parameters are easily recovered. For instance approximately 20% of all λs are fully recovered using an enumeration effort of 2^{16}.



(a) $\lambda \in \mathbb{F}_{2^{128}}$ (b) $\lambda \in \mathbb{F}_{2^{256}}$

Figure 4.12: Distribution of log rank of λ for the non-optimized schoolbook multiplication.

The use of simulations in Section 4.5 showed a direct connection between the required security level (i.e. the key rank) and the SNR, which is reflected in Figure 4.9 and 4.10. Namely, these figures allow inferring quickly a rough estimate on the security level by simply computing the SNR of the device. However, there is no direct formal link between these simulations and the real experiments. Among possible reasons, we note that, while the SNR is constant for all operations in the simulations, it is quite variable in experimental traces. There are also some additional leakage points that can appear in real traces that are not captured by simulations. Yet, we can informally discuss what we observe for the two cases. For the naive schoolbook implementation, the SNRs of the leakages range between 0.1 and 0.4. Interestingly, the resulting ranks showed in Figure 4.12 are fairly in line with the simulations of Figure 4.9 and 4.10 for these SNR values. That is, the security graphs from the simulations show that the implementation is likely to be vulnerable for such SNR values, which is what we observe in practice.

4.6.3 Operand caching multiplication

The second part of our experiment consists in applying the same process of D&C attack against an optimized implementation, namely the operand-caching one. We recall that this method is a variant of the

LIM that reduces the number of memory accesses to gain in efficiency, thereby reducing the leakages on the operands and on the result. Thus, it seems to constitute a natural and efficient way to lessen the impact of the previous attack.

For each byte of λ, we perform the same systematic steps as for the non-optimized schoolbook implementation. On Figure 4.13, we plot the evolution of the guessing entropies for the different bytes in function of the number of dimensions after PCA compression. While for the non-optimized multiplication all bytes of λ are loaded the same number of times, here they are color coded according to the number of times they are loaded. First, we observe the same behavior as for the previous implementation when it comes to the number of PCA components: we require a large number of components to decrease the guessing entropies. Next, we clearly notice the impact of the amount of loads on the guessing entropy. That is, less loads tend to lead to a higher entropy in comparison with the non-optimized implementation. It is clear that the limited leakages in this case do not allow reaching guessing entropies below 20 candidates, which is significantly more than on the naive schoolbook multiplication.

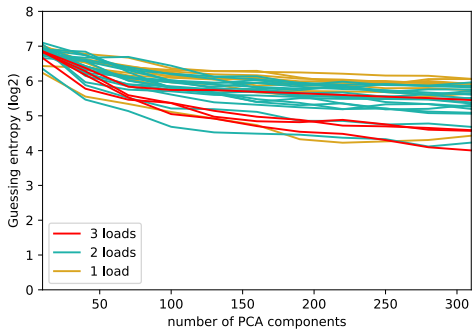
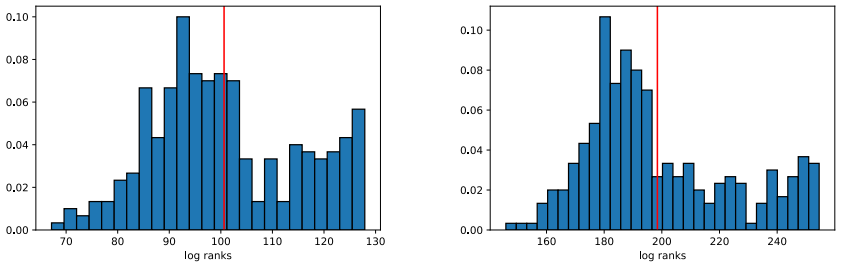


Figure 4.13: Logarithm of the guessing entropy of the 32 bytes of λ as function of the number of PCA components for the optimized operand-caching multiplication.

Next, we plot the distribution of the log ranks for both 128- and 256-bit λ in Figure 4.14. We observe that due to the optimizations applied for the operand caching the leakages are minimized, leading to significantly different results compared to the non-optimized implementation studied in the previous subsection. For the 128-bit size parameter, we obtain an average log rank of 100 bits, and for the 256-bit parameter we obtain an average log rank of 200 bits. The entropy of λ is only marginally reduced

and its value cannot be recovered with a reasonable enumeration effort.



(a) $\lambda \in \mathbb{F}_{2^{128}}$ (b) $\lambda \in \mathbb{F}_{2^{256}}$

Figure 4.14: Distribution of log rank of λ for the operand caching multiplication.

Similarly to the schoolbook method, we informally discuss the link between the simulated results and the real attack against the operand-caching implementation. This time, most of the SNR values for the POI are below 0.05, and memory operations (1, 2 or 3 loads) achieve a maximum SNR ≈ 0.25. The security graphs from the simulations indicate that the rank of the secret parameter is likely to be out of reach even with strong enumeration efforts. Interestingly, this is also what we observe in practice, as we were not able to break the implementation using the operand-caching method.

Overall, the comparison of the non-optimized multiplication and the optimized one highlights how point randomization can be made quite robust against nearly worst-case adversaries without much performance overheads: as clear from [86], the additional cost of point randomization is limited for (already expensive) ECC implementations. That is, a few design considerations and optimizations suffice to reduce the overall leakages that can be exploited by an adversary, such that single-trace horizontal attacks become impractical. We note that, as usual in experimental side-channel attacks, these results obviously depend on the measurement setup and the preprocessing applied to the traces, which can possibly be improved. Yet, the conclusion that limited SNR reductions are enough to secure point randomization, and that simple optimizations (that are motivated by general performance concerns) are good for this purpose, should hold in general.

4.7 Projective coordinates system comparison

In our preliminary analysis, we focused on a homogeneous projective point representation. This section investigates the use of a different coordinate system, namely Jacobian coordinates, and the possible consequences regarding side-channel resistance. For this purpose, using the LRPM and its extensions proposed in Section 4.4, we further extend our analysis by comparing the randomization in homogeneous and Jacobian projective coordinates. For the former, we consider both cases where the full (x, y) representation is randomized (leading to two exploitable multiplications λx and λy) and the x-only case [60] (where a single multiplication λx can be exploited). In the Jacobian case, the coordinates randomization requires four multiplications to compute λ^2, λ^3, $\lambda^2 x$, and $\lambda^3 y$. Here, we consider again two cases: one where a generic multiplication is used to perform the squaring λ^2, and one where the squaring is implemented efficiently. The results of our evaluations are depicted on Figure 4.15 where MI upper bounds are plotted as a function of the SNR.

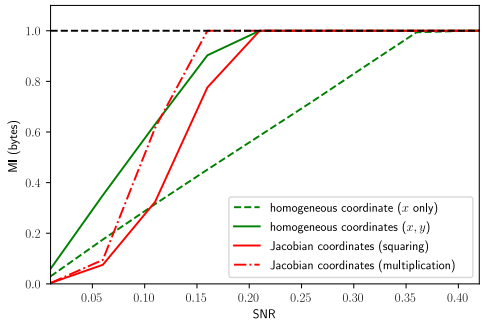


Figure 4.15: Comparison of SASCA on homogeneous and Jacobian projective coordinates randomization.

.

First, and as previously shown, we confirm that for homogeneous coordinates, since λx and λy are independent operations when targeting λ, the overall information is simply doubled. Secondly, when it comes to Jacobian coordinates, using a squaring instead of a multiplication to compute λ^2 leads to improved security. Regarding the comparison, while the coordinate system used is typically dictated by the ECSM's overall performance, the results lead to interesting conclusions. They suggest that for low SNR (< 0.1), Jacobian coordinates randomization is more resistant to worst-case side-channel adversaries than homoge-

neous coordinates. For $\mathsf{SNR} > 0.1$, the x-only homogeneous projective randomization is the best option, followed by the Jacobian coordinates randomization with an optimized squaring operation.

Regarding the comparison between D&C attacks and SASCA for the specific case of Jacobian coordinates, we note that the D&C adversary is more limited with respect to the exploitable intermediates, as he can only target one of the four multiplications at a time. The most optimal D&C target operation is either $\lambda^2 x$ or $\lambda^3 y$, but not both simultaneously because the D&C method cannot link them, since each word of λ^3 depends on multiple words of λ^2 (and both on multiple words of λ). The attack's success would thus be similar to targeting λx in the case of homogeneous coordinates shown in Figure 4.15. This would widen the gap between D&C attacks and SASCA for $\mathsf{SNR} > 0.1$ for Jacobian coordinates. As a result, and as opposed to the homogeneous coordinates case, we see that the SNR impacts the method of choice for a security evaluation. Indeed, for high SNR, using a D&C strategy is less effective than a worst-case analytical attack. The factor graph of Jacobian coordinates randomization is extremely large and makes SASCA impractical for evaluations. However, we are able to efficiently bound the information recovered on the secret after a BP-based attack on this randomization using the LRPM as demonstrated in Section 4.5 and shown on Figure 4.15.

4.8 Worst-case evaluation of 32-bit implementations

In this section, we translate our previous investigation on 8-bit implementations to the practically-relevant case of 32-bit devices. First, we emphasize that implementing an actual attack against a 32-bit implementation is much more challenging. While modeling 32-bit leakage is feasible by using a linear-regression-based approach (similarly to the horizontal attack on the Montgomery ECSM by Poussier, Zhou and Standaert [134] detailed in Chapter 3, Section 3.2), exploitation on the other hand is very demanding since it would require enumerating 2^{32} values, leading to large computation and storage efforts. Some workarounds are possible by trading computational complexity for algorithmic noise.

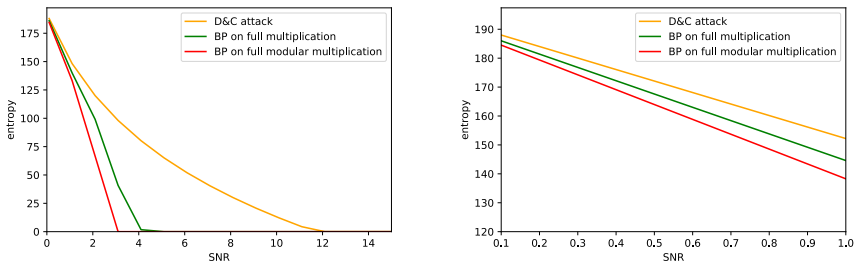
For instance, in [94] it is suggested to calculate the probability vector on the HW of an 8-bit variable A after observing the HW leakage of the 32-bit value $A|B|C|D$ by simply summing over all possible HW values as follows:

$$\Pr(\mathsf{HW}(A) = h_a) = \sum_{h_b,h_c,h_d} \Pr(\mathsf{HW}(A|B|C|D) = h_a + h_b + h_c + h_d) \cdot \Pr(\mathsf{HW}(B) = h_b) \cdot \Pr(\mathsf{HW}(C) = h_c) \cdot \Pr(\mathsf{HW}(D) = h_d)$$

where the HWs of B, C and D follow the same probabilities as the HW of a uniform and unknown 8-bit value. The information that can be extracted on $\mathsf{HW}(A)$ from the leakage of the full register is heavily impacted by the algorithmic noise due to targeting 8 bits out of 32. Another option that we suggest is to turn a D&C attack requiring 2^{32} enumerations into a BP-based attack requiring a 2^8 enumeration. This is done by decomposing every 32-bit operation into a factor graph of 8-bit variables and multiple operations, where all intermediates, besides the inputs and outputs to the 32-bit operation, are not leaking. However, this method would also suffer greatly from the added algorithmic noise, and would not correspond to an optimal strategy. Any adversary targeting 8 bits out of 32 is bounded by the optimal attacker able to perform 2^{32} enumerations and capable of performing BP on the full factor graph of the randomization.

Since we focus on worst-case security, we analyze the information obtained by a strong attacker able to perform attacks on 32-bit leakages directly. For that purpose, we use the LRPM. As shown in Section 4.5 on Figures 4.9 and 4.10, the LRPM provides a lower bound on the guessing entropy after the attack. Based on this observation, we bound the guessing entropy when targeting a 32-bit implementation for different attacks (D&C and SASCA) and for different SNR levels. The results of this analysis are plotted on Figure 4.16 for high (left) and low (right) SNR cases. First, we note that when the SNR is low the gain of analytical strategies over D&C strategies is still very limited as for the 8-bit case. But more interestingly, we observe that implementations of the point randomization on 32-bit devices are able to resist worst-case adversaries as shown by the pessimistic lower bounds on the guessing entropy on Figure 4.16. Concretely, for an $\mathsf{SNR} \approx 0.9$, as measured in a recent attack targeting an STM32F405 [94], it is possible to achieve excellent concrete security even against attackers exploiting all the possible leakage of the point randomization. This is a positive result that suggests that securing 32-bit ECC implementations (e.g., on ARM devices) against very powerful attackers might be feasible in practice.

However, we can clearly see a gap between both methods when the SNR is roughly above 2. This is explained by the fact that a higher SNR is necessary to recover information on the targeted operations for the



(a) High SNR (b) Low SNR

Figure 4.16: Entropy lower bound comparison of SASCA on the full graph of the field multiplication, and the D&C attack for a 32-bit implementation with $\lambda \in \mathbb{F}_{2^{256}}$.

32-bit case. In this high SNR case, operations that are only exploited by the BP attack, such as additions, start to provide a decent amount of information, leading to a gap between the two methods. Since the implementation was already broken for lower SNR values in the 8-bit case, this gap could not appear. This means that, for high SNR values, using a D&C strategy does not translate into a close counterpart to a worst-case attacker for a 32-bit implementation, as opposed to the 8-bit case.

4.9 Conclusion

In this chapter, we investigated the security of the point randomization countermeasure with respect to worst-case attackers. We showed how to apply SASCA when targeting point randomization and we additionally adapted a recent and efficient evaluation methodology for SASCA using the LRPM to this asymmetric use-case. Using this model, we showed that for realistic noise levels on 8-bit devices, there is almost no gain in mounting a complex SASCA making use of all the available information compared to a simpler D&C attack that can be complemented with enumeration. As a result, using simulations, we estimated the required SNR needed by implementations in small embedded devices to be secure. Somewhat surprisingly, we observed that the point randomization technique can be implemented quite securely even in 8-bit devices.

We then performed practical experiments against basic and optimized implementations to further illustrate the impact of performance optimizations: while the naive implementation is shown to be broken,

the optimized one leads to ranks that are not reachable with enumeration, without any additional countermeasures. Finally, we provided guessing entropy lower bounds for challenging attacks on 32-bit implementations which again confirm the resistance of point randomization against side-channel attacks. Interestingly, our results indicate that the point randomization on 32-bit devices provides excellent concrete security against worst-case adversaries. This leads to the positive conclusion that secure 32-bit ECC implementations might be feasible in practice, if we can achieve similar high security for the following ECSM.

Chapter 5

Amplified Bitslice Masking and Shuffling

It is well established by the cryptographic community that security on embedded devices requires strong or even a combination of strong countermeasures. It was also suggested that combining complementary countermeasures such as masking and hiding is more favorable [112]. However, side-channel countermeasures typically entail a significant overhead in terms of time and randomness. For instance, masking typically leads to a quadratic cost increase. This raises the important question of whether combining countermeasures leads to practical and cost-effective security improvements. In this respect, the combination of masking and shuffling was previously investigated by Rivain, Prouff and Doget [140]. Assuming a shuffled execution of η independent operations (e.g., Sboxes), each of them masked with d shares, they showed that for a sufficient noise variance σ^2, the complexity to attack the corresponding implementation grows in $\mathcal{O}\big(\eta \cdot (\sigma^2)^d\big)$.

In this chapter, we revisit this popular combination, and study its application in the bitslice implementation setting and its resistance against worst-case side-channel adversaries. Our contributions in this respect are the following. First, as a consolidating contribution, in Section 5.2 we describe an improvement of the multivariate attacks against shuffled implementations proposed by Veyrat-Charvillon et al. [164] that we use in our security evaluations. In particular, we show that in this case and for high noise levels D&C attacks are sufficient to approach worst-case attacks. Our main contribution is presented in Section 5.3. By systematically analyzing the combination of masking and shuffling, we obtain an efficient solution to improve it. This is achieved by shuffling noisy shares rather than noisy tuples of shares, which leads to amplifying exponentially in the masking order the impact of the noise emulated by

shuffling. Then, in Section 5.4, we investigate the applicability of our solution. In this respect, we consider the bitslice strategy introduced by Biham [21] and applied in various recent works [47, 73, 16, 72, 15, 5]. We show that under sufficient noise, it is better to prioritize masking over shuffling, i.e. fill the registers with the shares and shuffle the remaining independent operations. Finally, in Section 5.5 we focus on the challenge of implementing algorithmic countermeasures on real devices that only offer minimal physical noise. While the work of Rivain et al. [140] relied on correlation-based attacks, and concluded that shuffling is still useful even when the physical noise is limited, recent research has highlighted how such evaluations are lacking and overestimate the security of an implementation, when the adversary can model the leakage of the randomness [28]. We consider an experimental evaluation on an ARM Cortex-M3 and show that profiled attacks can cancel the impact of shuffling.

5.1 Preliminaries

In this section we provide some introductory material and background about the tools we use in the remainder of this chapter, in addition to a description of the countermeasures we investigate and their theoretical security impacts.

5.1.1 Mutual information

We introduced the MI in Chapter 2, Section 2.3.5 as a metric to assess the information extracted from the leakage on a secret key or intermediate related to the secret. In addition, the MI can be used to evaluate the effectiveness of both countermeasures and attack strategies. In particular, the relation between the MI of a secret variable X and its leakage L, and the number of traces N required to recover the value of X from L is given by [53]:

$$N \geq \frac{cst}{\mathsf{MI}(X; L)} \tag{5.1}$$

The small constant cst depends on the entropy $\mathrm{H}(X)$ of the targeted variable, the desired success rate of the attack [53, 46] and marginally on the correlation between the target variable's possible candidates [154]. Choosing $cst = \mathrm{H}(X)$ corresponds to a success rate $> 50\%$.

The MI usually corresponds to a worst-case adversary with a perfect modeling of the leakage. When we consider imperfect modeling or suboptimal attacks (with respect to the worst-case adversary), we use the MI counterpart also introduced in Section 2.3.5, the PI.

5.1.2 Masking

We recall the definition of masking from Section 2.4. Boolean masking decomposes a secret variable a into an encoding or a tuple $\mathbf{a} = [a_0, a_1, \ldots, a_{d-1}]$ of d uniformly distributed shares such that $a = a_0 \oplus a_1 \oplus \ldots \oplus a_{d-1}$. As a result of this representation, and as long as all the d shares remain independent, a is not revealed unless at least d leakage samples are exploited. Under this masking scheme, linear operations are performed share-wise, however non-linear operations, such as multiplications, require more attention since they do not process the shares independently. To tackle this issue Ishai, Sahai and Wagner [88] suggested to perform secure multiplications as described by Algorithm 11, where $\otimes$ denotes the multiplication in $\mathbb{F}_2$. This algorithm requires $d \cdot (d-1)/2$ random bits r.

Algorithm 11 ISW multiplication.

Input: $\boldsymbol{a}$ with $a = \sum_{i=0}^{d-1} a_i$ and $\boldsymbol{b}$ with $b = \sum_{i=0}^{d-1} b_i$.
Output: $\boldsymbol{c}$ with $c = \sum_{i=0}^{d-1} c_i$ and $c = a \otimes b$.

for i in $[0, \ldots, d-1]$ **do**
  $c_i \leftarrow a_i \otimes b_i$
for i in $[0, \ldots, d-1]$ **do**
  for j in $[i+1, \ldots, d-1]$ **do**
    $r \leftarrow \{0, 1\}$
    $c_i \leftarrow c_i \oplus (r \oplus (a_i \otimes b_j))$
    $c_j \leftarrow c_j \oplus (r \oplus (a_j \otimes b_i))$

A bound on the number of traces required to break a masked implementation with d shares was given by Duc, Faust and Standaert [53]:

$$N \geq \frac{cst}{\prod_{i=0}^{d-1} \mathsf{MI}(A_i; L)} \tag{5.2}$$

This bound is subject to two conditions: the independence of the shares as formalized by probing security [88], such that the security order d is maintained, and additionally sufficiently high side-channel noise such that the overall MI on a decreases exponentially in d.

5.1.3 Shuffling

Shuffling consists in randomizing the order of the cipher's operations based on a random permutation such that it becomes difficult to identify

relevant leakage [83]. This countermeasure leverages independent operations that are indistinguishable and can be permuted. Algorithm 12 shows an example of a shuffled execution. The same operation op is applied to η elements $(y^i)_{0\leq i<\eta}$ (e.g., the AES Sbox operation applied to the η=16 Sbox inputs). The first step of shuffling the η operations is to generate a random permutation $\boldsymbol{\theta}$ over the set $\{0, 1, \ldots, \eta-1\}$. The set of all possible permutations over $\{0, 1, \ldots, \eta-1\}$ is denoted by $\boldsymbol{\Theta}$. In the following, we always assume that $\boldsymbol{\theta}$ is picked randomly and uniformly from $\boldsymbol{\Theta}$. The size of the permutation is given by π. For now, it is equal to η, but it will be seen in the next sections of this chapter, that there are also cases where $\pi < \eta$. Another interesting case corresponds to the addition of dummy operations to increase the size of the permutation such that $\pi > \eta$ but we do not discuss this case in this chapter. The execution of the operations op is performed out-of-order, and the index s that defines the order in which the elements $(y^i)_{0\leq i<\eta}$ are processed to yield $(z^i)_{0\leq i<\eta}$ is chosen based on $\boldsymbol{\theta}$. Concretely, at iteration c the index of the processed input element y^s is equal to θ_c.

Algorithm 12 Shuffled execution.

Input: $y^0, y^1, \ldots, y^{\eta-1}$
Output: $z^0, z^1, \ldots, z^{\eta-1}$ such that $\forall i, 0 \leq i < \eta, z^i = \mathsf{op}(y^i)$.

$\boldsymbol{\theta} \xleftarrow{\$} \boldsymbol{\Theta}$ ▷ Perm. generation
for c in $[0, \ldots, \eta-1]$ **do**
  $s \leftarrow \theta_c$
  $z^s \leftarrow \mathsf{op}(y^s)$

Shuffling increases the data complexity of an attack by a linear factor equal to η, if a full entropy permutation is used, i.e. generated randomly and uniformly from the set of all possible permutations over the same elements [83, 164]. The relation between the number of attack traces N, the $\mathsf{MI}(X; L)$ of variable X knowing its leakage L without shuffling and the number of shuffled variables η among which X is shuffled is given by the equation below. This relation holds only if the side-channel noise is high enough to hide the permutation.

$$N \geq \frac{cst \cdot \eta}{\mathsf{MI}(X; L)} \tag{5.3}$$

5.2 Improved attacks on shuffling

In the following, we provide an improvement of the D&C attacks proposed by Veyrat-Charvillon et al. [164] at Asiacrypt 2012. To do so, we first describe the leakage observed from a shuffled implementation. Following Algorithm 12, for each index c we observe the leakage of $s = \theta_c$, that we call the *permutation leakage* and denote by l^{θ_c}. Additionally, we observe the *data leakage* coming from the manipulation of the input (resp. output) and that we denote by $l^{y^{\theta_c}}$ (resp. $l^{z^{\theta_c}}$).

A profiled maximum-likelihood attack requires the estimation of the leakage's Probability Density Function (PDF). The optimal way to compute the leakage PDF of a shuffled implementation is given by the next equation:

$$\mathsf{f}(\boldsymbol{l}|\boldsymbol{y}) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \Pr(\boldsymbol{\theta}) \cdot \mathsf{f}(\boldsymbol{l}|\boldsymbol{\theta}, \boldsymbol{y}) \tag{5.4}$$

where f denotes leakage PDFs, $\boldsymbol{y}$ the input data vector $(y^s)_{0\leq s<\eta}$, and $\boldsymbol{l}$ the concatenation of the permutation and data leakage vectors $(l^{\theta_c})_{0\leq c<\eta}$ and $(l^{y^{\theta_c}})_{0\leq c<\eta}$. However, based on the size of the permutation π, the sum over all possible permutations can be computationally prohibitive, e.g., for $\pi = \eta = 16$ the number of modes of this mixture distribution is already $16! \approx 2^{44.2}$. As a result, more computationally efficient approaches have been proposed, at the cost of a possible information loss. One option is to ignore the permutation leakage and simply assume a uniform probability for all the permutation elements leading to the following PDF for each input y^s:

$$\mathsf{f}(\boldsymbol{l}|y^s) = \frac{1}{\eta}\sum_{c=0}^{\eta-1} \mathsf{f}(l^{y^{\theta_c}}|y^s) \tag{5.5}$$

In this case, we do not make use of any information coming from the permutation leakage. To remedy this, Veyrat-Charvillon et al. [164] propose a more efficient alternative that exploits both the data and permutation leakage. We describe this attack in the next section.

5.2.1 The Asiacrypt2012 attack

The Asiacrypt2012 attack of Veyrat-Charvillon et al. [164] which targets shuffling is expressed by the following equation:

$$\mathsf{f}_{\mathsf{AC12}}(\boldsymbol{l}|y^s) = \sum_{c=0}^{\eta-1} \underbrace{\mathsf{w}_{\mathsf{AC12}}(\theta_c = s|\boldsymbol{l}^{\boldsymbol{\theta}})}_{\text{perm. leak.}} \cdot \underbrace{\mathsf{f}(l^{y^{\theta_c}}|y^s)}_{\text{data leak.}} \tag{5.6}$$

where $\mathsf{w}_{\mathsf{AC12}}(\theta_c = s|\boldsymbol{l}^{\boldsymbol{\theta}})$ is the weight assigned to a cycle or execution index c. Its goal is to indicate the likelihood that the targeted value y^s is manipulated at the cycle c. They propose many solutions to derive $\mathsf{w}_{\mathsf{AC12}}$ with different time and data complexities. We focus on the so-called *direct permutation leakages* (DPLeak in [164]), for which this weight is computed as:

$$\mathsf{w}_{\mathsf{AC12}}(\theta_c = s|\boldsymbol{l}^{\boldsymbol{\theta}}) = \frac{\mathsf{f}(l^{\theta_c}|\theta_c = s)}{\sum_{c'=0}^{\eta-1} \mathsf{f}(l^{\theta_{c'}}|\theta_{c'} = s)} \tag{5.7}$$

The last step to recover the probability on the full data vector $\boldsymbol{y}$ is to apply Bayes' theorem and normalize the probabilities according to:

$$\tilde{\Pr}_{\mathsf{AC12}}(\boldsymbol{y}|\boldsymbol{l}) = \prod_{s=0}^{\eta-1} \frac{\mathsf{f}_{\mathsf{AC12}}(\boldsymbol{l}|y^s)}{\sum_{y^{s^*}} \mathsf{f}_{\mathsf{AC12}}(\boldsymbol{l}|y^{s^*})} \tag{5.8}$$

As in Section 2.3.5, the $\sim$ denotes an imperfect leakage modeling. In this case, $\tilde{\Pr}_{\mathsf{AC12}}$ leads the attacker to exploiting only some PI from the leakage rather than the full MI. Additionally, we note that despite this attack is not summing over all the permutations, it is not a D&C one since the leakages of the permutation are still exploited jointly as expressed in the denominator in Equation 5.7.

5.2.2 Improvements description

In this part, we propose an improved attack strategy in comparison to the Asiacrypt2012 attack, which is also just as computationally efficient. This attack is very similar to the Asiacrypt2012 attack, but expresses the probability model in a different way to optimize the information extracted from the leakage. It is given by the following equation:

$$\tilde{\Pr}_{\mathsf{New}}(\boldsymbol{y}|\boldsymbol{l}) = \prod_{s=0}^{\eta-1} \sum_{c=0}^{\eta-1} \underbrace{\mathsf{w}_{\mathsf{New}}(\theta_c = s|l^{\theta_c})}_{\text{perm. leak.}} \cdot \underbrace{\frac{\mathsf{f}(l^{y^{\theta_c}}|y^s)}{\sum_{y^{s^*}} \mathsf{f}(l^{y^{\theta_c}}|y^{s^*})}}_{\text{data leak.}} \tag{5.9}$$

Our improvements in comparison to the Asiacrypt2012 attack are twofold. First, the weighted sum over c is not performed on continuous PDFs anymore but on the probabilities obtained after the application of Bayes' theorem, as reflected by the right term in Equation 5.9. This is the usual approach taken for advanced attacks such as SASCA [163] to extract information on intermediate values. Second, we notice that the weights in the sum estimated with Equation 5.7 depend on the full permutation leakage. Since each term in the sum corresponds to the

leakage at cycle c, we modified the weights such that they correspond instead to the probability that the index manipulated at cycle c is equal to s. This yields the following weight computation:

$$\mathsf{w}_{\mathsf{New}}(\theta_c = s|l^{\theta_c}) = \frac{\mathsf{f}(l^{\theta_c}|\theta_c = s)}{\sum_{s'=0}^{\eta-1} \mathsf{f}(l^{\theta_c}|\theta_c = s')} \tag{5.10}$$

5.2.3 Attack models comparison

In this section we compare the Asiacrypt2012 model $\tilde{\Pr}_{\mathsf{AC12}}$ that we refer to in the following as DPLeak AC12, and the improved model proposed in this thesis $\tilde{\Pr}_{\mathsf{New}}$ that we refer to as DPLeakNew.

5.2.3.1 Methodology.

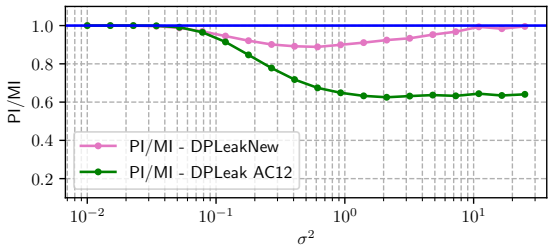
For the purpose of the comparison, we simulate a shuffled implementation according to Algorithm 12, following the distribution defined in Equation 5.4. In particular, our simulations are parametrized by the Gaussian noise variance σ^2 and the number of independent shuffled operations or intermediates represented by the vector $\boldsymbol{y} = [y^0, y^1, \ldots, y^{\eta-1}]$ such that $\pi = \eta$. Additionally, we assume that the leakages of the elements in $\boldsymbol{y}$ are independent, and compute the noise variance on the permutation leakage to match the same SNR as the data leakage.

In this simulated setting, we know the true leakage PDF and we can estimate the MI according to Equation 2.5 in Section 2.3.5. The MI captures the best worst-case attack possible against this shuffling example. Since, for larger permutations the estimation of the MI is prohibitive, we consider permutations of size $\pi \in \{2, 4, 6\}$. Then, for the same implementation examples, we estimate the PI using Equation 2.6 for both models given by Equation 5.8 and 5.9 respectively. We note that only the computation of the MI is intensive, while the PI estimates do not suffer from this limitation and can be computed for larger permutations. Since we assumed the independence of the elements in $\boldsymbol{y}$, we have that $\mathrm{H}(\boldsymbol{Y}) = \sum_{i=0}^{\eta-1} \mathrm{H}(Y^i) = \eta \cdot \mathsf{H}(Y^i)$. Finally, by considering the ratio between the PI values and the MI, we can assess how far each model is from the optimal attack enumerating all possible permutations.

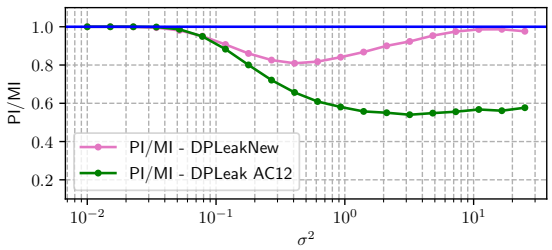
5.2.3.2 Results and discussion.

The results of the previously described simulations and IT ratios PI/MI are reported in Figure 5.1 as function of the noise variance. This figure highlights the gap between the optimal attack (corresponding to the MI), and the two alternative attacks based on the AC12 model and the new

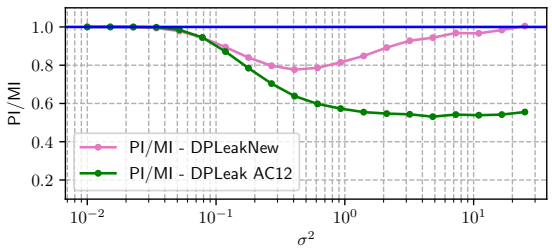
model in this chapter. Based on these results, we first observe that our new model improves over the AC12 one. Indeed, the PI for the DPLeak AC12 adversary is always and in most cases significantly lower than the PI for DPLeakNew. Second, we observe that for high noise levels, our new model offers a good approximation of the MI while the AC12 one suffers from a significant bias, which increases with the noise variance.



(a) $\eta = 2$, PI/MI



(b) $\eta = 4$, PI/MI



(c) $\eta = 6$, PI/MI

Figure 5.1: IT analysis of shuffled implementation in Algorithm 12.

We note that the DPLeakNew attack could possibly be further improved with analytical attacks such as [163]. However, Figure 5.1 shows that for high noise levels on which we will further focus, DPLeakNew is already very close to the worst-case attack, which is in line with the observation in Chapter 4, that such analytical attacks only lead to minor improvements when targeting ephemeral secrets such as a shuffling permutation, and in particular for high noise cases [8].

In the rest of this chapter, small-scale IT evaluations based on the worst-case MI as shown on Figure 5.1, will be quite systematically used in order to evaluate and compare different combinations of masking and shuffling. We therefore use this first application to discuss their main pros and cons as follows:

- On the negative side: (*i*) these small scale examples only correspond to attacks considering a representative leakage function, which is in not equivalent to proving security in general, and (*ii*) they do not directly apply to the typical sizes of concrete implementations (e.g., $\mathsf{H}(Y^i) = 8$ and $\eta = 16$ for the AES) since we cannot estimate the MI for such large parameters.
- On the positive side: (*i*) IT evaluations as we suggest have quite systematically been shown to be excellent indicators of the bounds that can be obtained in masking security proofs: see for example the sequence of papers [154, 155, 136, 52, 53] for an illustration, and (*ii*) the same holds (up to constant factors) for the extrapolation from small variables to larger ones, and the concrete attacks we propose do not suffer from computational limitations.

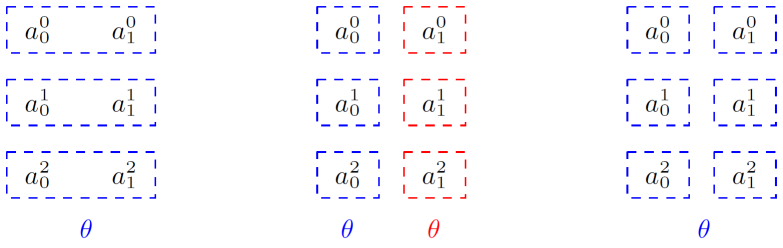
Since all our observations are confirmed for a few (growing) values of d and π, we therefore believe they provide a necessary first step towards the understanding of the combination of masking and shuffling which, in particular, improves over the one of Rivain et al. [140], both in terms of the security levels we claim and in terms of the considered attacks' coverage. Proving the bounds we observe in a formal model such as the RPM [52] is an interesting scope for further research.

5.3 Information theoretic analysis of combined masking and shuffling

A circuit of a cryptographic function masked to achieve $(d-1)^{th}$ order security is composed of two kinds of operations: linear ones that can be applied share-wise, and non-linear ones that require to securely mix the shares to maintain the security order. In the following, we investigate different ways to combine masking and shuffling for both linear and non-linear operations. We then analyze the relevant options with IT metrics. For now we focus on the high noise regime, where exploiting the permutation leakages does not improve the attacks [164]. The case of low noise regime will be later discussed based on a real-world case-study in Section 5.5.

5.3.1 Linear operations

Applying a linear operation op to a secret encoding $\boldsymbol{a}$ consists in applying op to all of its elements $(a_i)_{0\leq i<d}$ independently. Accordingly, the output encoding $\boldsymbol{b}$ is simply given by $b_i = \mathsf{op}(a_i), \forall i \in \{0, 1, \ldots, d-1\}$. The same holds when we have η such encodings $(\boldsymbol{b}^s)_{0\leq s<\eta}$. The challenge when combining masking and shuffling for linear layers is to identify in what order should pairs of indexes (i, s) be used to load a_i^s. We consider three possible shuffling configurations that can be applied to a linear layer, as summarized in Figure 5.2 where a color denotes a permutation and a box the pair(s) (i, s) that are accessed deterministically at each cycle of that permutation.



(a) Shuffling tuples. (b) Shuffling shares. (c) Shuffling everything.

Figure 5.2: Shuffling configurations of a linear encoding for $d = 2$ and $\eta = 3$.

5.3.1.1 Shuffling tuples.

A first straightforward possibility is to shuffle tuples of shares. Only the indexes s of the vectors are shuffled and the shares i of that index are accessed sequentially in order. Algorithm 13 is an example of such a combination and a graphical representation is given in Figure 5.2(a).

In terms of security, this shuffling and masking combination is similar to the case where we only shuffle an unmasked implementation such as shown by Algorithm 12. Indeed, because the shares are accessed in order, at the cycle c information on all the shares $a_i^{\theta_c}$ can be identified and combined together to get information on a^{θ_c}. Essentially, this countermeasure combination only impacts the information on a^{θ_c}, but does not impact the information on the individual shares. So similarly to a shuffled-only implementation, the security of this combination is linear

Algorithm 13 Masking and shuffling tuples.

Input: $\boldsymbol{a}$ such that $a^s = \sum_{i=0}^{d-1} a_i^s$ and $\mathsf{op}(\cdot)$
Output: $\boldsymbol{b}$ such that $b^s = \mathsf{op}(a^s) = \sum_{i=0}^{d-1} b_i^s$

$\boldsymbol{\theta} \xleftarrow{\$} \boldsymbol{\Theta}$ ▷ Perm. generation of size $\pi = \eta$
for c in $[0, \ldots, \eta - 1]$ **do**
  $s \leftarrow \theta_c$
  for i in $[0, \ldots, d - 1]$ **do**
    $b_i^s = \mathsf{op}(a_i^s)$

in the number of operations η on which we shuffle:

$$N^{\text{tuples}} \geq \frac{cst \cdot \eta}{\prod_{i=0}^{d-1} \mathsf{MI}_u(\boldsymbol{A}_i; \boldsymbol{L})} \tag{5.11}$$

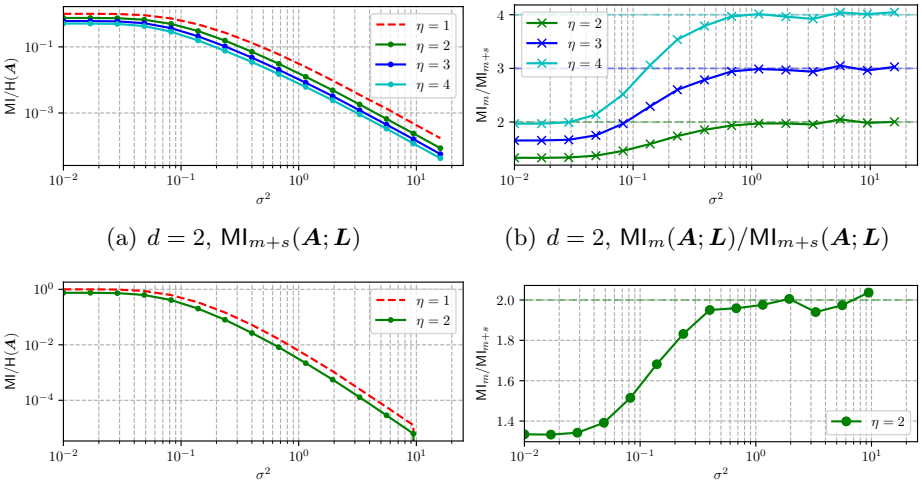
where the MI subscript u (for $u$nprotected) denotes the fact that the countermeasures have no affect on the individual shares' information.

This equation is confirmed by the IT analysis of Algorithm 13 given in Figure 5.3. On the left, we report the MI of a masked and shuffled implementation $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$, for various parameters d and η. On the right, we report the ratio between the MI of a masked-only implementation $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})$ and $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$. This ratio is equal to the increase of the (worst-case) attack data complexity that shuffling and masking provide compared to masking only. We first observe that, as expected from Equation 5.11, the gain is equal to the permutation size $\pi = \eta$ for sufficiently large noise variance. We additionally observe that this gain is independent of d, which confirms that there is no non-trivial interaction between shuffling and masking in this case. Indeed for $\eta = 2$, the gain is equal to 2 for the two and three shares implementations.

5.3.1.2 Shuffling shares.

A natural option to improve the interaction between masking and shuffling is to shuffle the shares of independent variables instead of their tuples. As illustrated in Figure 5.2(b) and described formally in Algorithm 14, it consists in processing the shares in order and shuffling over the vectors $\boldsymbol{a}_i^s$ for a fixed share index i. This option requires picking a fresh random permutation of size $\pi = \eta$ for each share index processing. As a result, the permutation is always applied on independent values that never involve the same encoding $\boldsymbol{a}^s$.

This approach is beneficial to side-channel security since to obtain information about a secret a^s, an adversary now has to retrieve at which



(a) $d = 2$, $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ (b) $d = 2$, $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})/\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$

(c) $d = 3$, $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ (d) $d = 3$, $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})/\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$

Figure 5.3: Shuffling tuples (Algorithm 13): IT analysis.

Algorithm 14 Masking and shuffling shares.

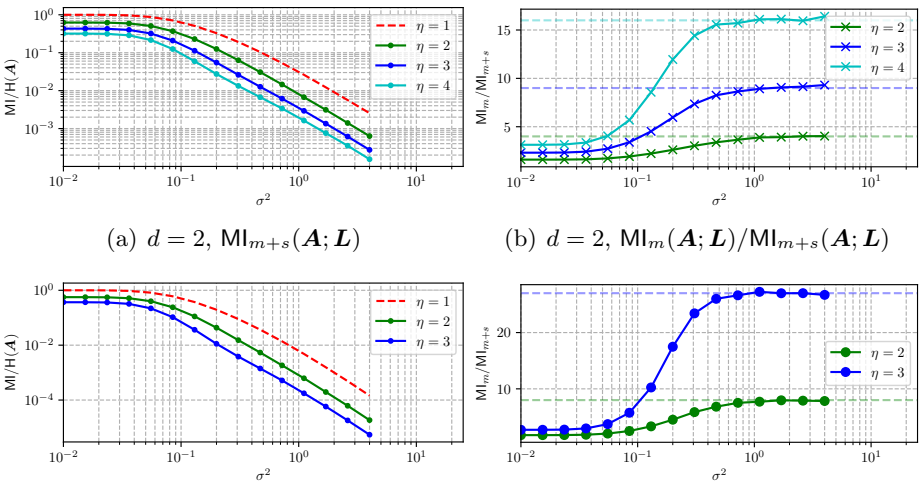
Input: $a^s = \sum_{i=0}^{d-1} a_i^s$ and $\mathsf{op}(\cdot)$
Output: $b^s = \mathsf{op}(a^s) = \sum_{i=0}^{d-1} b_i^s$

for i in $[0, \ldots, d-1]$ **do**
  $\boldsymbol{\theta} \xleftarrow{\$} \boldsymbol{\Theta}$ ▷ Perm. generation of size $\pi = \eta$
  for c in $[0, \ldots, \eta - 1]$ **do**
    $s \leftarrow \theta_c$
    $b_i^s = \mathsf{op}(a_i^s)$

cycles c the d shares a_i^s are manipulated. Without knowledge of the permutations (e.g., because of sufficiently noisy leakages as we assume for now), she succeeds for a share with probability $\frac{1}{\eta}$, and hence with probability $\left(\frac{1}{\eta}\right)^d$ for the d shares. As a result, the shuffling shares method can be interpreted as providing an increase of the noise on each share by a factor equal to the permutation size $\pi = \eta$. Masking then amplifies this emulated noise exponentially in d. The impact of this combination on the (worst-case) attack data complexity is therefore given by:

$$N^{\text{shares}} \geq \frac{cst}{\prod_{i=0}^{d-1} \mathsf{MI}_u(\boldsymbol{A}_i; \boldsymbol{L})/\eta} = \frac{cst \cdot \eta^d}{\prod_{i=0}^{d-1} \mathsf{MI}_u(\boldsymbol{A}_i; \boldsymbol{L})} \tag{5.12}$$

As before, this equation is confirmed by the systematic IT analysis of Algorithm 14 in Figure 5.4. We observe that for large noise, the $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ of the shuffled shares implementation is lower than the $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})$ of the masked only implementation by a factor η^d. For example, for $\eta = 4$ and $d = 2$, this ratio is $4^2 = 16$. For $\eta = 3$ and $d = 3$, this ratio is $3^3 = 27$. Therefore, by using d permutations of size $\pi = \eta$, this solution provides an exponential amplification of the noise generated by shuffling.



(a) $d = 2$, $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ (b) $d = 2$, $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})/\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$

(c) $d = 3$, $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ (d) $d = 3$, $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})/\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$

Figure 5.4: Shuffling shares (Algorithm 14): IT analysis.

5.3.1.3 Shuffling everything.

The two previous options shuffled either over the shares or over the tuples. A last solution is to shuffle jointly all the shares of all tuples corresponding to all the pairs (i, s) by using a single permutation on $\pi = d \cdot \eta$ elements. This combination is illustrated in Figure 5.2(c) and formally defined in Algorithm 15.

Algorithm 15 Masking and shuffling everything.

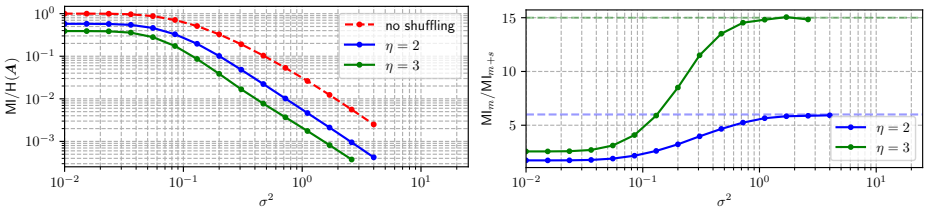
Input: $a^s = \sum_{i=0}^{d-1} a_i^s$ and $\mathsf{op}(\cdot)$
Output: $b^s = \mathsf{op}(a^s) = \sum_{i=0}^{d-1} b_i^s$

$\boldsymbol{\theta} \xleftarrow{\$} \boldsymbol{\Theta}$ ▷ Perm. generation of size $\pi = d \cdot \eta$
for c in $[0, \ldots, (d \cdot \eta) - 1]$ **do**
  $(i, s) \leftarrow (\theta_c \% d, \theta_c / d)$
  $b_i^s = \mathsf{op}(a_i^s)$

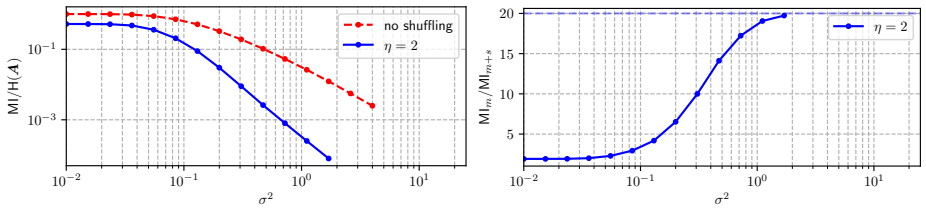
In terms of side-channel security, recovering information about a secret a^s, without knowledge of the permutation, now requires the adversary to consider the leakage of all d shares of all η tuples. To recover a^s, this implies to find the cycles c where the d pairs (i, s) with $0 \leq i < d$ are used. To do so, she chooses the first share out of the set of $d \cdot \eta$ shuffled values. Since d of these values correspond to a share a_i^s, she succeeds with probability $\frac{d}{d \cdot \eta}$. The second share can be guessed correctly with probability $\frac{d-1}{d \cdot \eta - 1}$ by excluding the first selected share. Eventually, the adversary can obtain information on a^s if she selects correctly all the d shares which happens with probability $\prod_{i=0}^{d-1} \frac{d-i}{d \cdot \eta - i} = \frac{1}{\binom{d \cdot \eta}{d}}$. Hence, the attack data complexity is growing as:

$$N^{\text{everything}} \geq \frac{cst \cdot \binom{d \cdot \eta}{d}}{\prod_{i=0}^{d-1} \mathsf{MI}_u(\boldsymbol{A}_i; \boldsymbol{L})} \tag{5.13}$$

Once again, we confirm this equation with an IT analysis of a simulated implementation of Algorithm 15 in Figure 5.5. For large noise, the IT ratio $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L}) / \mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ is equal to $\binom{2 \cdot 2}{2} = 6$ for $d = 2$ and $\eta = 2$, equal to $\binom{2 \cdot 3}{2} = 15$ for $d = 2$ and $\eta = 3$ and finally equal to $\binom{3 \cdot 2}{3} = 20$ for $d = 3$ and $\eta = 2$. This confirm the combinatorial security amplification of this combination of shuffling and masking.



(a) $d = 2$, $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ (b) $d = 2$, $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})/\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$



(c) $d = 3$, $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ (d) $d = 3$, $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})/\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$

Figure 5.5: Shuffling everything (Algorithm 15): IT analysis.

5.3.1.4 Discussion.

We conclude from the previous analyses and experiments that $N^{\text{tuples}} \leq N^{\text{shares}} \leq N^{\text{everything}}$. However, these successive improvements do not come for free but at the cost of more or larger permutations. Hence, they raise the question of which option provides the best cost vs. security trade-off, which we will discuss in Section 5.4. For this purpose, a preliminary step is to generalize our results from linear operations to non-linear ones, which we tackle in the next section. We note that Rivain et al. used the *shuffling everything* method for their linear layers (and derived a correlation-based bound for this purpose), but they only used the *shuffling tuples* one for the AES Sboxes [140]. To the best of our knowledge, the *shuffling shares* method is new. As will be shown next, it is also the one leading to the best cost vs. security trade-off. In particular, it allows avoiding the imbalance between the security levels of linear and non-linear operations, which is caused by the use of two types of shuffling, and forced Rivain et al. to artificially increase the permutation size of the non-linear layers by using dummy operations.

5.3.2 Non-linear operations

The second building block for common cryptographic circuits are non-linear operations. We will consider the standard ISW multiplication [88] for this purpose, described in Algorithm 11, but the conclusions in this

chapter also apply to other masking schemes based on a similar structure. In this case, all the pairs (i, j) have to be accessed in order to compute the cross products $a_i \otimes b_j$. We assume a setting where η independent ISW multiplications $c^s = a^s \otimes b^s$, with $0 \leq s < \eta$, have to be computed. In order to perform such operations, it implies that all the triplets (s, i, j) have to be accessed to compute all the $a_i^s \otimes b_j^s$ cross-products. Next, we list different ways to shuffle the computation of these triplets and discuss the different security levels they lead to.

We note that such combinations of masking and shuffling for non-linear layers are more complex than for linear ones: there are more possibilities to be considered and their security is sometimes hard to assess. We therefore start by presenting the simplest solution of *shuffling tuples* used by Rivain et al. [140]. We then introduce a generic taxonomy of shuffling configurations which allows describing the design space of the masking + shuffling combinations, and we illustrate this taxonomy with the *shuffling tuples* approach. Finally, we prune this design space and focus on a number of solutions that can be viewed as the counterparts of the *shuffling shares* and *shuffling everything* approaches previously described for linear operations. We also argue why this pruning is practically relevant.

5.3.2.1 Shuffling tuples.

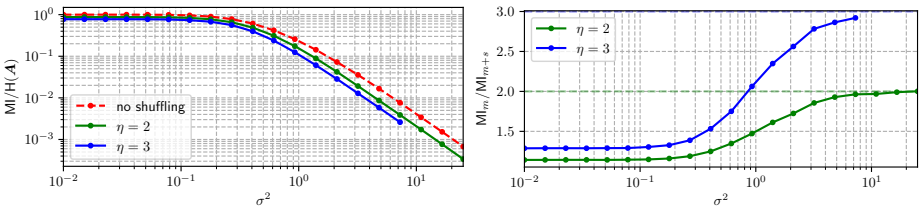
This first method is similar to the *shuffling tuples* for linear layers. It is presented in Algorithm 16, where only the accesses to the tuples $\boldsymbol{a}^s$ and $\boldsymbol{b}^s$ are shuffled with a permutation of size $\pi = \eta$. Then the pairs (i, j) are accessed deterministically within these tuples. The sequence of operations in Algorithm 16 is slightly different than in Algorithm 11. Namely both i and j range from 0 to $d-1$ and the first loop on $a_i \otimes b_i$ is omitted. The constraints on $r_{i,j}^s$ make these two algorithms equivalent. Similarly to the shuffling tuples for linear layers, this allows an adversary to obtain information on the unshared secrets a^s, b^s and c^s with probability $\frac{1}{\eta}$. The resulting security guarantee is then linear with the size of the permutation similarly to Equation 5.11. This is confirmed by our IT analysis of Algorithm 16 reported in Figure 5.6. There, the ratio between the MI of a masked only ISW multiplication $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})$ and the one of shuffled and masked ISW multiplication $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ is always equal to η. This combination is a straightforward adaptation of the Rivain et al. [140] implementation of masked and shuffled AES SubBytes to the case of ISW multiplications.

Algorithm 16 Shuffling tuples ISW (configuration $(1^s, 0^i, 0^j)$).

Input: inputs $\{\boldsymbol{a}^0, \boldsymbol{a}^1, \ldots, \boldsymbol{a}^{\eta-1}\}$ and $\{\boldsymbol{b}^0, \boldsymbol{b}^1, \ldots, \boldsymbol{b}^{\eta-1}\}$ and randomness $r^s_{i,j}$ defined as: $\forall s, \forall i, \forall j$, such that $i < j$, $r^s_{i,j} \xleftarrow{\$} \{0,1\}$ and $r^s_{j,i} = r^s_{i,j}$ and $r^s_{i,i} = 0$.

Output: outputs $\{\boldsymbol{c}^0, \boldsymbol{c}^1, \ldots, \boldsymbol{c}^{\eta-1}\}$ such that $\forall s \in \{0, 1, \ldots, \eta-1\}, c^s = a^s \otimes b^s$.

$\boldsymbol{\theta}^s \xleftarrow{\$} \boldsymbol{\Theta}^s$ ▷ permutation over η elements
for s in $\boldsymbol{\theta}^s$ **do**
  for i in $[0, \ldots, d-1]$ **do**
    for j in $[0, \ldots, d-1]$ **do**
      $c^s_i \leftarrow c^s_i \oplus r^s_{i,j} \oplus (a^s_i \otimes b^s_j)$



(a) $d = 2$, $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ (b) $d = 2$, $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})/\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$

Figure 5.6: Shuffling ISW multiplication tuples (Algorithm 16): IT analysis.

5.3.2.2 Shuffling more.

As in the previous section investigating linear operations, the next step is to shuffle over more operations in order to reach a better security gain. For example, one natural goal would be to amplify the effect of shuffling with masking so that a gain factor of η^d can be maintained for full implementations across linear and non-linear operations. For this purpose, we first describe all the possible shuffling and masking combinations which handle the indexes s, i and j independently, with the next shuffling configurations.

Shuffling + masking configurations. At a high level, we describe all the possible combinations of masking and shuffling with Algorithm 17 along with a notation of the configurations of the form $(x^\alpha, x^\beta, x^\gamma)$. Algorithm 17 is composed of three nested loops where each loop is responsible for shuffling (or not) one of the indexes s, i or j. In the configuration $(x^\alpha, x^\beta, x^\gamma)$, the superscripts correspond to the index manipulated by the loop and the position in the triplet corresponds to the order of the loops. For example, the first superscript designates the outermost loop and the third one the innermost loop. Additionally, the x value is a bit set to 1 if the loop is shuffled and 0 otherwise. We note that swapping the indexes i and j leads to the same gadget since the multiplication is commutative. As a result, the configuration $(0^s, 0^i, 0^j)$ denotes an unshuffled implementation, and the previous *shuffling tuples* solution given in Algorithm 16 corresponds to the configuration $(1^s, 0^i, 0^j)$, where only the outer loop on s is shuffled. Finally, the Greek letters in Algorithm 17, namely $\alpha \in A, \beta \in B$ and $\gamma \in \Gamma$, are uniquely set to s, i or j. The corresponding capital letter is the set of values that the index can take. For example, if $\alpha = s$ then $A = [0, \ldots, \eta - 1]$ and if $\alpha = i$ or $\alpha = j$ then $A = [0, \ldots, d - 1]$. Hence, $\boldsymbol{\theta}^\alpha \xleftarrow{x^\alpha} \boldsymbol{\Theta}^\alpha$ is a fresh permutation of the elements in A if x^α is set to one and is equal to A in order otherwise.

Pruning the configuration space. Based on the previous configurations, there are 6 possible ways to order the loops and, for each of these orderings, there are 2^3 possibilities of shuffling. This leads to a total of 48 cases. In order to reduce the number of configurations to investigate in detail, we first notice that in Algorithm 17, shuffling the indexes i or j never leads to a security improvement. Indeed, shuffling on i (resp. j) only means shuffling the accesses to the shares a_i^s (resp. b_j^s). Because in an encoding $\boldsymbol{a}^s$ of a^s, the position of the shares has no effect on security, an adversary can obtain information on a^s by observing leakages on all the shuffled shares a_i^s without being impacted by the shuffling

Algorithm 17 Generic shuffled & masked ISW multiplications.

Input: inputs $\{\boldsymbol{a}^0, \boldsymbol{a}^1, \ldots, \boldsymbol{a}^{\eta-1}\}$ and $\{\boldsymbol{b}^0, \boldsymbol{b}^1, \ldots, \boldsymbol{b}^{\eta-1}\}$, shuffling configuration $(x^\alpha, x^\beta, x^\gamma)$ with $\alpha, \beta, \gamma \in \{s, i, j\}$ and randomness $r^s_{i,j}$ defined as: $\forall s, \forall i, \forall j$, such that $i < j$, $r^s_{i,j} \leftarrow \{0,1\}$ and $r^s_{j,i} = r^s_{i,j}$ and $r^s_{i,i} = 0$.

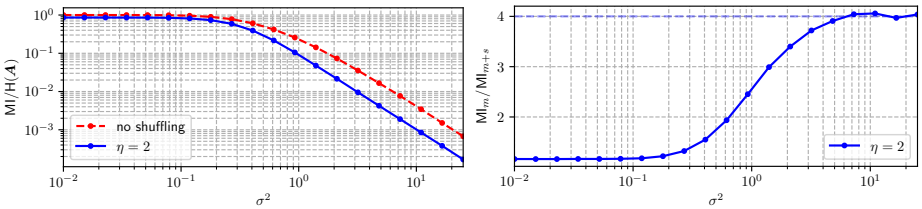
Output: outputs $\{\boldsymbol{c}^0, \boldsymbol{c}^1, \ldots, \boldsymbol{c}^{\eta-1}\}$ such that $\forall s \in \{0, 1, \ldots, \eta-1\}, c^s = a^s \otimes b^s$.

for α in $\boldsymbol{\theta}^\alpha \xleftarrow{x^\alpha} \boldsymbol{\Theta}^\alpha$ **do** ▷ one permutation over A
  for β in $\boldsymbol{\theta}^\beta \xleftarrow{x^\beta} \boldsymbol{\Theta}^\beta$ **do** ▷ $|A|$ permutations over B
    for γ in $\boldsymbol{\theta}^\gamma \xleftarrow{x^\gamma} \boldsymbol{\Theta}^\gamma$ **do** ▷ $|A| \cdot |B|$ permutations over Γ
      $c^s_i \leftarrow c^s_i \oplus r^s_{i,j} \oplus (a^s_i \otimes b^s_j)$

of i (resp. j). We further note that in the high-noise regime (that we assume for now), this observation also holds when we shuffle i and j. In this case, the cross-products are shuffled and information on each of the a^s_i involved in these cross-products is obtained from the η observations of the shuffled $a^{\boldsymbol{\theta}^s}_i$. But information on a^s can still be recovered from the input/output tuples of the multiplication, without being impacted by the shuffling of i. So this configuration makes the information of the cross-products harder to exploit, while it is known that the information provided by these cross-products is dominated by the information of the multiplications' input/output tuples when the noise is large, as shown by Cassiers and Standaert [32] in the LRPM. Therefore, we next limit our investigations to configurations with 0^i and 0^j.

This reduces the set of combinations to 3 possibilities. We next list them and discuss their security. The first one is $(1^s, 0^i, 0^j)$ and corresponds to the aforementioned *shuffling tuples* (Algorithm 16) for which the security impact is only linear in the size of the permutation. The second configuration is $(0^i, 1^s, 0^j)$ (resp. $(0^j, 1^s, 0^i)$). For this configuration the security of the inner loop variable b_j (resp. a_i) differs from the one on the outer loop variable a_i (resp. b_j) and the security of the inner loop variable is similar to the *shuffling tuples* option, which is not desirable. Finally, the third configuration is $(0^i, 0^j, 1^s)$. It is similar to the *shuffling shares* of linear layers, where the permutation is applied to $\pi = \eta$ independent elements. In this case, every loading of the input shares a^s_i and b^s_j as well as the updates of c^s_i are shuffled among the η independent ISW multiplications. As a result, the information on each of these operations is reduced by a factor η that is later amplified by masking. Therefore, it provides an exponential gain factor η^d as in

Algorithm 5.12. This gain is confirmed by the IT analysis depicted in Figure 5.7 for $d = 2$ and $\eta = 2$, where the ratio between $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})$ and $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ for high enough noise is of $2^2 = 4$.



(a) $d = 2$, $\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$ (b) $d = 2$, $\mathsf{MI}_m(\boldsymbol{A}; \boldsymbol{L})/\mathsf{MI}_{m+s}(\boldsymbol{A}; \boldsymbol{L})$

Figure 5.7: Shuffling ISW shares (Algorithm 17, config. $(0^i, 0^j, 1^s)$): IT analysis.

5.3.2.3 Shuffling everything-like.

A last natural question is to know whether it is possible to obtain a better security, similar to the one obtained for the *shuffling everything* of linear layers. For these operations, the improvement was obtained by shuffling jointly all the pairs (i, j) instead of independently. For the ISW multiplications, this can be done by merging some of (or all) the loops in Algorithm 17. To list the possible combinations when two loops are merged, we use Algorithm 18 along with notations of the form $(x^{\alpha,\beta}, x^{\gamma})$. For example, $(x^{i,j}, x^s)$ means that the outer loop is a merge of the loops on i and j and hence over d^2 elements. More precisely, in Algorithm 18 $\boldsymbol{\theta}^{\alpha,\beta}$ is a permutation over the set $A \times B$ (where $\times$ denotes the Cartesian product). Alternatively, all loops can be merged together into a single one operating on $d^2 \cdot \eta$ elements. We denote this configuration as $(x^{s,i,j})$ and detail it in Algorithm 19.

Based on these configurations, a first observation is that loops on i and j must be merged together in order to avoid the same asymmetry issues as in the previous *shuffling shares* approach. This reduces the possibilities of merging loops to three combinations that are $(1^s, 1^{i,j})$, $(1^{i,j}, 1^s)$ and $(1^{s,i,j})$. However, all these options induce a non-uniform permutation of the output shares. Figure 5.8 illustrates this effect for various parameters d and η. There, the y-axis is the probability that a given valid output share (in this case c_0^0) is generated during the t-th iteration of the algorithm. The x-axis is the iteration t that is normalized by the total number of iterations $\eta \cdot d^2$. To explain this non-uniformity, we stress that one output share c_0^0 is valid once it has been updated d

Algorithm 18 Generic shuffled masked ISW multiplications with 2 loops merged.

Input: inputs $\{\boldsymbol{a}^0, \boldsymbol{a}^1, \ldots, \boldsymbol{a}^{\eta-1}\}$ and $\{\boldsymbol{b}^0, \boldsymbol{b}^1, \ldots, \boldsymbol{b}^{\eta-1}\}$, shuffling configuration $(x^{\alpha,\beta}, x^{\gamma})$ with $\alpha, \beta, \gamma \in \{s, i, j\}$ and randomness $r^s_{i,j}$ defined as: $\forall s, \forall i, \forall j$, such that $i < j$, $r^s_{i,j} \leftarrow \{0,1\}$ and $r^s_{j,i} = r^s_{i,j}$ and $r^s_{i,i} = 0$.
Output: outputs $\{\boldsymbol{c}^0, \boldsymbol{c}^1, \ldots, \boldsymbol{c}^{\eta-1}\}$ such that $\forall s \in \{0, 1, \ldots, \eta-1\}, c^s = a^s \otimes b^s$.

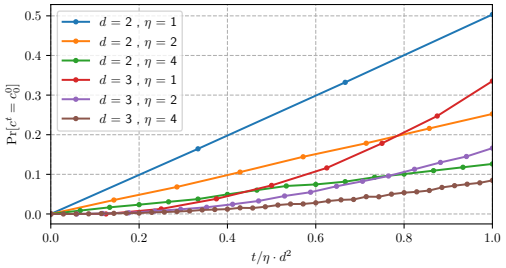
for α, β in $\boldsymbol{\theta}^{\alpha,\beta} \xleftarrow{x^{\alpha,\beta}} \boldsymbol{\Theta}^{\alpha,\beta}$ **do**
  for γ in $\boldsymbol{\theta}^{\gamma} \xleftarrow{x^{\gamma}} \boldsymbol{\Theta}^{\gamma}$ **do** ▷ $|A|$ permutations over B
    $c^s_i \leftarrow c^s_i \oplus r^s_{i,j} \oplus (a^s_i \otimes b^s_j)$

Algorithm 19 Generic shuffled masked ISW multiplications with all loops merged.

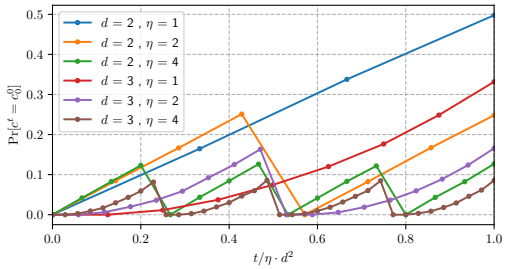
Input: inputs $\{\boldsymbol{a}^0, \boldsymbol{a}^1, \ldots, \boldsymbol{a}^{\eta-1}\}$ and $\{\boldsymbol{b}^0, \boldsymbol{b}^1, \ldots, \boldsymbol{b}^{\eta-1}\}$, shuffling configuration $(x^{\alpha,\beta,\gamma})$ with $\alpha, \beta, \gamma \in \{s, i, j\}$ and randomness $r^s_{i,j}$ defined as: $\forall s, \forall i, \forall j$, such that $i < j$, $r^s_{i,j} \leftarrow \{0,1\}$ and $r^s_{j,i} = r^s_{i,j}$ and $r^s_{i,i} = 0$.
Output: outputs $\{\boldsymbol{c}^0, \boldsymbol{c}^1, \ldots, \boldsymbol{c}^{\eta-1}\}$ such that $\forall s \in \{0, 1, \ldots, \eta-1\}, c^s = a^s \otimes b^s$.

for α, β, γ in $\boldsymbol{\theta}^{\alpha,\beta,\gamma} \xleftarrow{x^{\alpha,\beta,\gamma}} \boldsymbol{\Theta}^{\alpha,\beta,\gamma}$ **do**
  $c^s_i \leftarrow c^s_i \oplus r^s_{i,j} \oplus (a^s_i \otimes b^s_j)$

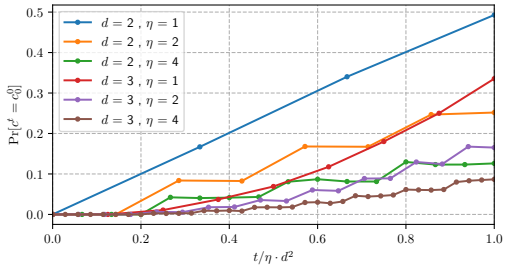
times in an ISW multiplication. Therefore, during the first few iterations it is unlikely that the current operation is the d-th update of c_0^0 as it is computed during the last few iterations. As a result, in all the plots of Figure 5.8, the probability to generate c_0^0 globally increases with t, with some differences depending on the shuffling configuration.



(a) Configuration $(1^{s,i,j})$.



(b) Configuration $(1^s, 1^{i,j})$.



(c) Configuration $(1^{i,j}, 1^s)$.

Figure 5.8: Probability to generate output share c_0^0 at cycle t when merging ISW loops.

We note that such non-uniform permutations of the output shares may offer some (even good) level of security. Yet, they suffer from the significant drawback that their analysis would require analyzing complex permutation biases that are specific to the configurations and the parameters d and η. We therefore rule out these options in our following

investigations for usability reasons.

5.3.3 Investigation summary

Table 5.1 summarizes the different combinations of masking and shuffling we considered, when the secret vectors are of size η. The table reports the security gain factor compared to a masked-only implementation (i.e. $\mathsf{MI}_m/\mathsf{MI}_{m+s}$), the size of the permutation(s) used (i.e. π) and the number of fresh permutation(s) needed (i.e. #perm.). As mentioned above, the *shuffling everything* approach cannot be straightforwardly applied to non-linear layers. Hence, we next focus on the *shuffling shares* solution which allows the same exponential security gain for both linear and non-linear layers, enabling balanced designs (and evaluate the *shuffling tuples* option for comparison purposes).

	Linear layer			Non-linear layer		
	$\mathsf{MI}_m/\mathsf{MI}_{m+s}$	π	#perm.	$\mathsf{MI}_m/\mathsf{MI}_{m+s}$	π	#perm.
shuffling tuples	η	η	1	η	η	1
shuffling shares	η^d	η	d	η^d	η	d^2
shuffling everything	$\binom{d\cdot\eta}{d}$	$d\cdot\eta$	1	?	?	?

Table 5.1: Summary of the masking + shuffling combinations.

5.4 Cost vs. security for bitslice combined masking and shuffling

In the previous section, we have discussed how to amplify the impact of shuffling by combining it with masking, for both linear and non-linear operations, and analyzed the different ways to achieve that. In this section, we focus on the practical implementation of the *shuffling tuples* and *shuffling shares* methods, with a focus on 32-bit software platforms, and we compare them to a masked-only implementation. In particular, we study Algorithm 17 with configurations $(1^s, 0^i, 0^j)$, $(0^i, 0^j, 1^s)$ and $(0^s, 0^i, 0^j)$ in the context of bitslice software implementations. We first detail the randomness that is required by such implementations. Then, we describe the parameters that influence their execution time and their security level. Finally, we propose concrete performance evaluations and leverage them in order to discuss general guidelines for combinations of masking and shuffling that best trade performance and security.

As a preliminary remark, we mention that when protecting a cryptographic implementation with shuffling, a preliminary (cipher-specific)

challenge is to find independent operations that can be executed in parallel. In order to keep the following discussions independent of the primitive to protect, we therefore consider a general use case where we aim to implement #AND ISW multiplications in parallel. As will be clear next, it allows us to draw general conclusions about how masking and shuffling should be combined (i.e. which of the countermeasures should use the available parallelism in priority), which can then be directly translated into concrete guidelines for implementing actual ciphers (or possibly modes of operations, in case they offer additional levels of parallelism).

5.4.1 Randomness requirements

The randomness required to combine masking and shuffling is composed of two terms. The first one is due to masking and remains independent of the shuffling parameter π. For the masked (only) ISW multiplication from Algorithm 11, $d \cdot (d-1)/2$ random bits per bitwise multiplication are needed. This means that #AND times more random bits are needed in our context. The second term is due to shuffling and corresponds to the randomness needed to generate the permutation(s) $\boldsymbol{\theta}$. Precisely, a permutation on η elements can be generated with $\eta \cdot \log_2 \eta$ random bits [164]. When combined with masking by *shuffling tuples*, a single of these permutations must be generated, as reported in Table 5.1. When combined with masking by *shuffling shares*, a total of d^2 of these permutations must be generated. As a result, the *shuffling shares* strategy requires a total amount of random bits given by:

$$\underbrace{\left(\#\text{AND} \cdot \frac{d \cdot (d-1)}{2}\right)}_{\text{masking}} + \underbrace{\left(d^2 \cdot \eta \cdot \log_2 \eta\right)}_{\text{shuffling}} . \tag{5.14}$$

5.4.2 The bitslice masking + shuffling design space

We now introduce the different parameters that influence the execution time as well as the security level of masked and shuffled bitslice implementations. These parameters are summarized in Table 5.2, where the first block contains parameters that are under control of the implementers and the second one contains parameters that depend (partially) on the software platform used.

We next detail these parameters and their interactions. First, we recall that bitslicing is a software programming technique that represents an algorithm (e.g., a block cipher) with Boolean operations [21]. It takes advantage of the parallelism enabled by bitwise instructions which are

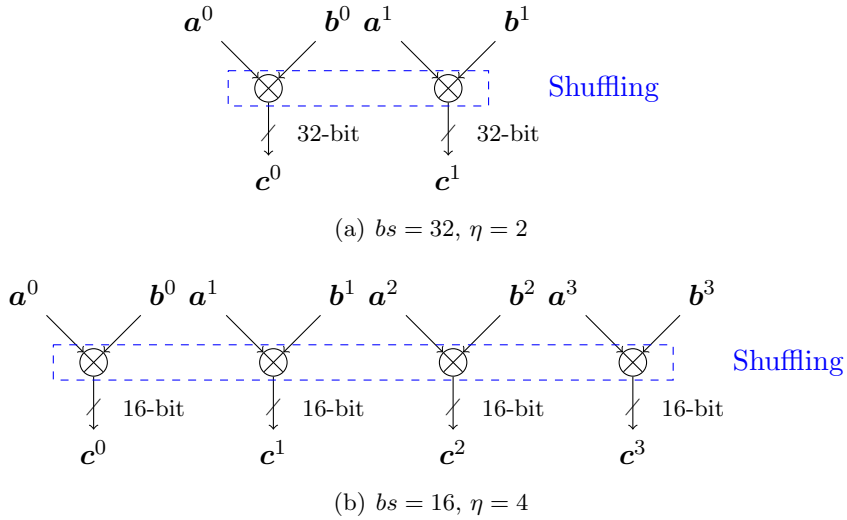
Parameter	Description
#AND	Number of masked bitwise AND to perform
bs	Effective size of the registers
$\pi = \eta$	Size of the permutations such that $\eta = \frac{\#\text{AND}}{bs}$
d	The number of shares
r	Number of cycles to generate 32 random bits
$\mathsf{MI}_u(\boldsymbol{A}^i; \boldsymbol{L})$	Mutual information on a single unshuffled shared bit

Table 5.2: Summary of parameters for bitslice masking and shuffling.

available in most (all) modern micro-controllers. For example, it is possible to place 32 bits in a register $\boldsymbol{a}$, 32 bits in a register $\boldsymbol{b}$, and to obtain the 32 bits corresponding to $\boldsymbol{a} \oplus \boldsymbol{b}$ in a single instruction. This strategy is particularly appealing in the context of masking where multiple bitwise ISW multiplications can be applied in parallel. Several works have shown how it can be efficiently used to protect block cipher implementations at arbitrary security orders [73, 16, 15]. Next, we observe that in contrast with bitslice implementations that favor parallelism, shuffling rather applies to serialized independent operations: the more such operations, the larger the size of the permutation and therefore the impact of shuffling. As a result, the main question when combining bitslice masking and shuffling is whether one should favor serialization or parallelism.

In order to answer this question, we will analyze the impact of the "effective size" of the register, denoted as bs, which is the parameter reflecting the trade-off between serialization and parallelism. It is defined as the number of useful bits in a single register, and therefore corresponds to the number of Boolean operations that are parallelized. The maximum value of bs is given by the physical register size. Therefore, on a 32-bit software platform, we have $bs \leq 32$. Yet, the designer could also reduce bs to increase the permutation size η that is equal to $\#\text{AND}/bs$. For example, Figure 5.9 illustrates two options in the masking + shuffling design space for #AND=64. The first option (Figure 5.9(a)) is to maximize the parallelism with $bs = 32$ and so $\eta = 2$. A second option (Figure 5.9(b)) is to decrease the parallelism with $bs = 16$ and to increase the permutation size up to $\eta = 4$.

The other parameters that we need to evaluate the performance vs. security trade-off of masked and shuffled implementations are the randomness cost and the noise level. Precisely, and as described in Section 5.4.1, both masking and shuffling require randomness. Hence, the



(a) $bs = 32$, $\eta = 2$

(b) $bs = 16$, $\eta = 4$

Figure 5.9: Different options in the design space for #AND=64.

throughput at which random bits are generated has an impact on the cycle count. We introduce the parameter r that is the latency in cycles needed to generate 32 random bits once requested. As for the noise level, we quantify it with the amount of information that is available on a single share of an unprotected implementation $\mathsf{MI}_u(\boldsymbol{A}^i; \boldsymbol{L})$.

5.4.3 Performance evaluation and discussion

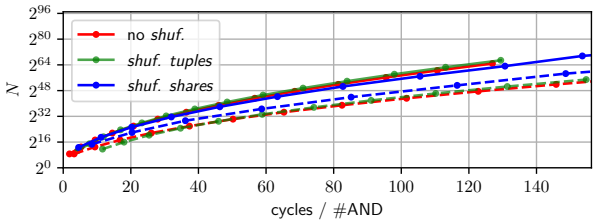
In order to evaluate the security vs. performance trade-off of the *shuffling tuples* and *shuffling shares* options, we complement the previous security evaluations by measuring the total cycle count of protected implementations running on a 32-bit ARM Cortex-M3 with the design parameters from Table 5.2. Precisely, when only masking we use a bitslice implementation of Algorithm 11 that is repeated η times to perform #AND bitwise secure multiplications. The security level is then given by Equation 5.2. For the *shuffling tuples* strategy, a similar approach is used but the η ISW multiplications are performed out of order following the generation of a single permutation. The security level is then given by Equation 5.11. For the *shuffling shares* strategy, its security level is given by Equation 5.12. We stress that the comparative value of our information theoretic analyses is due to the fact that the constants they imply are independent of the masking and shuffling parameters (i.e. they only depend on the size of the target intermediate variable and success rate). We also recall that they are only valid under a sufficient noise.

The concrete investigation of a low-noise case study is given in the next section.

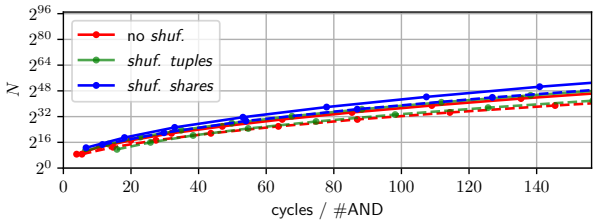
The resulting execution time vs. security curves are reported in Figure 5.10 for #AND=128 and in Figure 5.11 for #AND=512. The x-axis is the number of cycles used to perform the secure multiplications normalized by #AND. The y-axis reports the data complexity of the worst-case attack. Each data point is for a different masking order d, with the leftmost being $d = 2$ and increasing with steps of one when moving to the right. The red curves are for masked-only implementations, green ones for *shuffling tuples* and blue ones for *shuffling shares*. Continuous lines are for $bs = 32$, hence using the full physical register. The dashed curves are for $bs = 16$, hence doubling the serialization.

We draw two general conclusions regarding the combination of masking and shuffling from Figures 5.10 and 5.11. First, the choice of taking $bs = 32$ is always better than taking $bs = 16$. That is for a fixed execution time (x-axis), the resulting attack data complexity is always larger for $bs = 32$ than for $bs = 16$, and this trend holds for lower bs values. It implies that a designer should first favor the parallelization of the masked computations he has to perform, and shuffle what is left to serialize. One interesting corollary of this observation is that (for high enough noise levels as we consider), the use of dummy operations in order to increase the amount of operations to shuffle does not pay off: increasing the number of shares is a better strategy. Second, the combination of masking and shuffling benefits from a larger #AND. For example, if 80 cycles are spent per bitwise multiplication, the resulting security is about 2^{40} for #AND=128 for *shuffling shares* according to Figure 5.10(b). For the same number of clock cycles per bitwise multiplication, having #AND=512 provides a security around 2^{50} according to Figure 5.11(b).

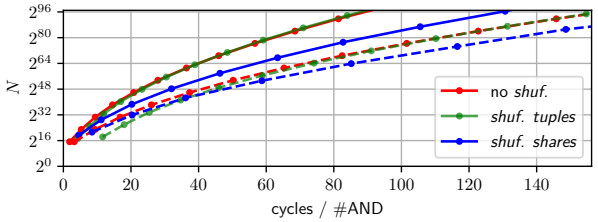
Masking only vs. shuffling tuples. By comparing the masked-only implementation with the *shuffling tuples* strategy, we observe that *shuffling tuples* does not bring a significant gain. That is, for a fixed performance level (i.e. value of the x-axis), *shuffling tuples* at best brings a marginal improvement. This can be explained by the fact that *shuffling tuples* is always slower than masking only due to the permutation generation, while only bringing a gain of η in attack complexity. Increasing this security gain would require to reduce bs, and so to favor serialization. But as shown above, this degrades the time vs. security trade-off which rather pushes for masking with maximum parallelism.



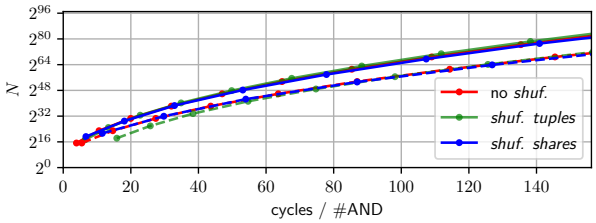
(a) $\mathsf{MI}_u(\boldsymbol{A}_i, \boldsymbol{L}) = 0.05$, $r = 8$



(b) $\mathsf{MI}_u(\boldsymbol{A}_i, \boldsymbol{L}) = 0.05$, $r = 64$

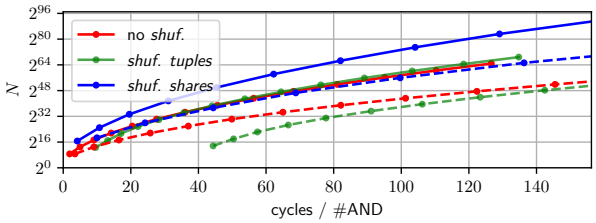


(c) $\mathsf{MI}_u(\boldsymbol{A}_i, \boldsymbol{L}) = 0.005$, $r = 8$

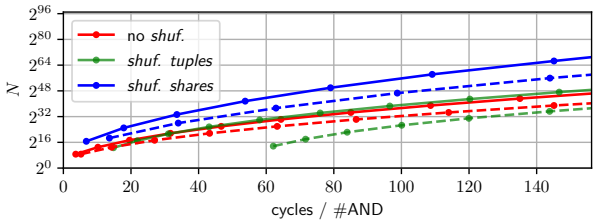


(d) $\mathsf{MI}_u(\boldsymbol{A}_i, \boldsymbol{L}) = 0.005$, $r = 64$

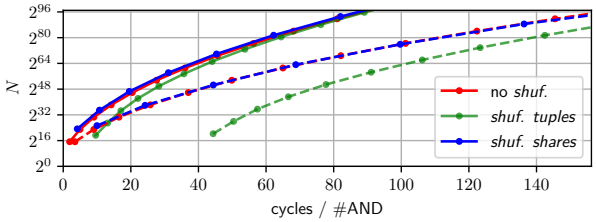
Figure 5.10: Cycles vs. security for 128 bitwise ANDs for various noise levels and randomness cost. Continuous are for $bs = 32$ and dashed ones for $bs = 16$.



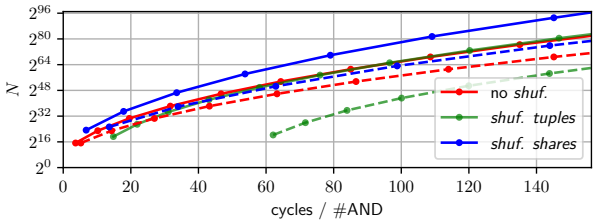
(a) $\mathsf{MI}_u(\boldsymbol{A}_i, \boldsymbol{L}) = 0.05$, $r = 8$



(b) $\mathsf{MI}_u(\boldsymbol{A}_i, \boldsymbol{L}) = 0.05$, $r = 64$



(c) $\mathsf{MI}_u(\boldsymbol{A}_i, \boldsymbol{L}) = 0.005$, $r = 8$



(d) $\mathsf{MI}_u(\boldsymbol{A}_i, \boldsymbol{L}) = 0.005$, $r = 64$

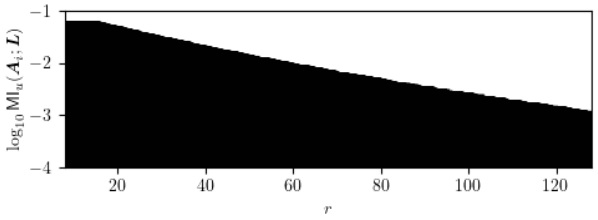
Figure 5.11: Cycles vs. security for 512 bitwise ANDs for various noise levels and randomness cost. Continuous are for $bs = 32$ and dashed ones for $bs = 16$.

Masking only vs. shuffling shares. Interestingly, the conclusion is more shaded when considering the *shuffling shares* approach. In this case, the interest of combining countermeasures essentially depends on the randomness cost r, noise level $\mathsf{MI}_u(\boldsymbol{A}^i; \boldsymbol{L})$ and amount of independent operations available #AND. That is, for expensive randomness, relatively low noise (still sufficient for the countermeasures to be effective) and high #AND, *shuffling shares* is the best solution. Whenever decreasing #AND or the randomness cost, or increasing the noise level, this advantage vanishes and masking-only can become the best option (e.g., in Figure 5.10(c)). We summarize this design space with Figure 5.12, where the x-axis is the randomness cost and the y-axis is the noise level $\mathsf{MI}_u(\boldsymbol{A}^i; \boldsymbol{L})$. Black areas represent contexts where masking alone is more efficient than *shuffling shares* in order to reach a given security target (here 64-bit). White areas represent contexts where *shuffling shares* is more efficient. Black areas are for cheap randomness and high noise levels (and lie in the bottom left). By increasing the number of independent multiplications #AND (and accordingly η), this area is reduced and the masking + shuffling approach becomes beneficial.

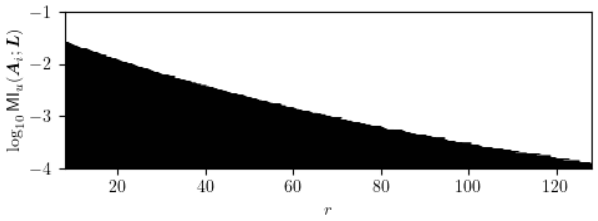
5.5 Real-world shuffling analysis

Based on the previous evaluations, we can conclude that the new (*shuffling shares*) combination of masking and shuffling that we propose in this work can indeed be an interesting asset for the design of side-channel secure implementations. It could in particular be quite effective to protect bitslice designs based on large permutations where a lot of parallel operations can be easily identified. Yet, these evaluations have been performed under the assumption of a sufficient noise level which may not be observed in practice. In this section, we therefore complement these analyses with a real-world evaluation of the masking + shuffling countermeasure implemented on a commercial MCU.

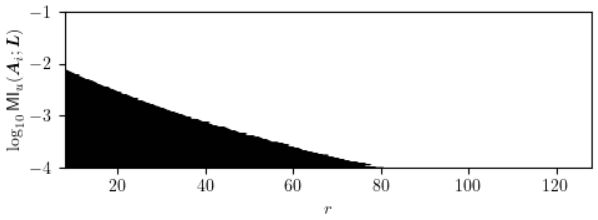
The motivation for this investigation comes from the intuition that shuffling could be an option to protect devices where the physical noise is not sufficient for masking to be directly effective. In order to clarify whether this expectation is founded, we propose a worst-case evaluation where the adversary has full knowledge and control of her target implementation during a profiling phase. In particular, she has knowledge of the randomness used to generate the permutations enabling profiled attacks against the shuffling as in Section 5.2. We conclude this section by briefly discussing how our observations evolve when relaxing the adversarial capabilities, in the spirit of a backwards evaluation [7].



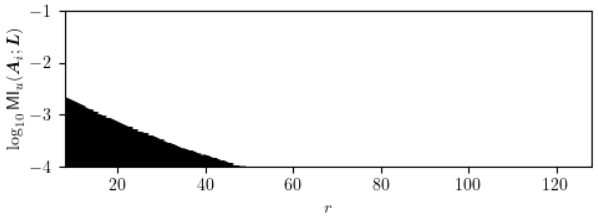
(a) #AND=128



(b) #AND=256



(c) #AND=512



(d) #AND=1024

Figure 5.12: Masking + *shuffling shares* design space for a 64-bit security target and $bs = 32$. Masking only is the most efficient solution in black areas.

5.5.1 Target and measurement setup

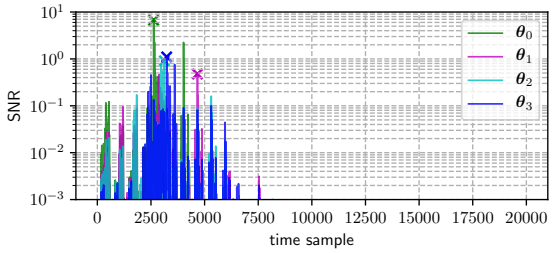
We investigate the implementation of ISW multiplications enhanced with the *shuffling shares* approach, executed on the 32-bit ARM Cortex-M3 of the STM32 VLDISCOVERY commercial board. In order to obtain clean measurements, we modified the board and removed all the decoupling inductances and capacitors. We used the available slot to add an 8 MHz external crystal and derived the maximum 24 MHz internal clock from it. In order to measure the side-channel leakage $\boldsymbol{L}$, we placed a current probe (i.e. the CT1 from Tektronix 1 GHz) on the dedicated jumper between the on-board power regulator and the MCU power pins. Eventually, we sampled this signal with a 12-bit resolution at a sampling rate of 500 MSamples/s thanks to a PicoScope 5244D. As shown next, the resulting setup allows collecting measurements with low noise.

5.5.2 Worst-case adversary

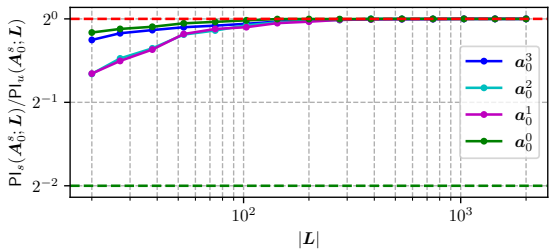
We evaluated this implementation with the adversary of Section 5.2 and more precisely with the model $\tilde{\mathsf{m}}_{\mathsf{New}}(\boldsymbol{y}|\boldsymbol{l})$ from Equation 5.9. For this purpose, we extract information on the secret variables using a standard Template attack in a principal subspace. Precisely, we estimate the leakage PDFs with Gaussian templates after dimensionality reduction using Linear Discriminant Analysis (LDA) [6].

The resulting leakage analysis is reported in Figure 5.13 for implementations with $\eta = 4$ and Figure 5.14 for $\eta = 16$. On the top figures, the x-axis is the time and the y-axis is the SNR (in log-scale) of the indexes of the permutation $\boldsymbol{\theta}_c$. We observe that the maximum value for each of them is around 1 and many other dimensions lead to an SNR two orders of magnitude lower (especially for $\eta = 16$). On the bottom figures, we report the number of dimensions of the leakage vector exploited by the adversary $|\boldsymbol{L}|$ on the x-axis and the y-axis reports (in log-scale) the information reduction per share. Namely it is the ratio between the PI of elements in the first share vector $\boldsymbol{a}_0$ for a $s$huffled implementation (i.e. $\mathsf{PI}_s(\boldsymbol{A}_0^c; \boldsymbol{L})$) and the one of an $u$nshuffled implementation (i.e. $\mathsf{PI}_u(\boldsymbol{A}_0^c; \boldsymbol{L})$). The green dashed curves represents the expected impact of the shuffling in case permutation indexes do not leak (as assumed in Section 5.3), namely $\frac{1}{\eta}$. The red dashed curve equals 1 and corresponds to a level of leakage such that shuffling is ineffective. As the number of dimensions exploited by the adversary increases, this ratio gets closer to one. For $\eta = 4$, it is stuck at 1, meaning that the adversary directly gains full knowledge of the permutation and is therefore not impacted by the shuffling. For $\eta = 16$, a similar behavior is observed but a larger

number of dimensions must be exploited. For $|\boldsymbol{L}| = 2,000$, the average ratio is equal to 0.92, meaning that the information per share is only reduced by 8%. For example, it implies that the gain factor of this combination of countermeasures with 8 shares is around 2 (while it would be 16^8 with a sufficient noise).



(a) SNR, $\eta = 4$

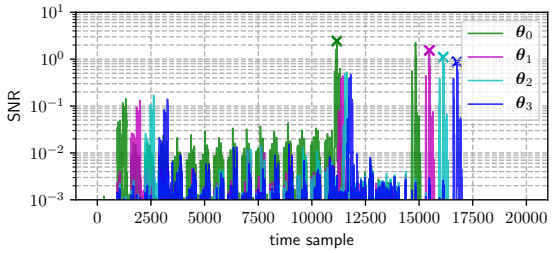


(b) $\mathsf{PI}_s(\boldsymbol{A}_0; \boldsymbol{L})/\mathsf{PI}_m(\boldsymbol{A}_0; \boldsymbol{L})$, $\eta = 4$

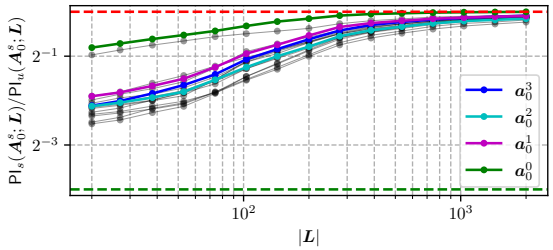
Figure 5.13: IT analysis of real permutation leakages with $\eta = 4$. Top: SNR in function of the time. Bottom: information loss in function of the # of dimensions exploited.

We note that these conclusions are based on a worst-case attack strategy. Non-profiled attacks unable to characterize (and therefore exploit) the permutation leakages may lead to a more positive conclusion (i.e. force the adversary to sum the noise variances of her shuffled operations). Yet, assuming that only such weaker attacks are possible appears as a risky strategy in view of recent progresses (e.g. using deep learning) where a worst-case information extraction is approached with limited knowledge of the target implementation [116].

Eventually, we insist that these experiments do not show that masking and shuffling cannot be implemented securely (the previous sections showed the opposite). What they show is that when the noise level provided by a leaking device is too low for masking to be effective, it is in general too low for shuffling to be effective as well. It may even be



(a) SNR, $\eta = 16$



(b) $\mathsf{PI}_s(\boldsymbol{A}_0;\boldsymbol{L})/\mathsf{PI}_m(\boldsymbol{A}_0;\boldsymbol{L})$, $\eta = 16$

Figure 5.14: IT analysis of real permutation leakages with $\eta = 16$. Top: SNR in function of the time. Bottom: information loss in function of the # of dimensions exploited.

more challenging to implement shuffling securely on low-cost embedded devices since the memory accesses it relies on can be more leaky than (for example) bitslice computations. So overall, it remains an important challenge to ensure a sufficient level of noise on embedded MCUs, so that masking, shuffling and their combination becomes effective. Reaching this goal with existing low-cost devices (similar to the ARM Cortex-M3 we analyzed) is an interesting research direction. Yet, the recurrent difficulties caused by low physical noise for the implementation of side-channel countermeasures also suggest that solving the issue at the technological level by guaranteeing a minimum level of intrinsic noise on security MCUs could be highly beneficial in terms of security vs. performance trade-off.

5.6 Conclusion

In this chapter, we improved the security, the applicability and the evaluation of an important combination of countermeasures: masking + shuffling. Concretely, we enhance the combination of masking and shuffling by introducing a solution that leads to an exponential security

impact. Additionally, We investigate this combination on a common embedded device and show that in practice, parallelism, which is the backbone of bitslice masked implementations, should always be favored over serialization, which is rather leveraged by shuffling. Moreover, our analyses also show that for high noise, cheap randomness or for a low level of serialization, masking is the best option. In the adverse settings, the *shuffling shares* solution suggested in this work can bring some improvements regarding the security vs. cost trade-off. Finally, we show experimentally that implementing shuffling securely against worst-case profiled attacks remains a challenge due to the limited noise on low-cost embedded devices.

Chapter 6

Conclusion and perspectives

High security guarantees for embedded cryptography can only be achieved by considering a worst-case adversary, as supported by the REASSURE backwards evaluation approach [7]. In this thesis, we aimed to improve the efficiency of side-channel security evaluations, while still maintaining high security confidence, and we tackled both symmetric and asymmetric cryptography examples. We achieved this by analyzing the information exploitation phase of side-channel attacks and considered evaluation shortcuts, D&C and worst-case attacks. In the following, we summarize our contributions and results and provide some directions for future research.

6.1 Shortcut Formulas for Horizontal Attacks on ECC

The ECSM is the central operation of ECC systems. In Chapter 3, we provided shortcut formulas for the success rate approximation of single-trace horizontal attacks applied to a regular binary ECSM. This shortcut formula is accurate under rapidly falsifiable assumptions. In our work, by analyzing the intermediate values computed during the ECSM and the side-channel behavior of a common micro-controller, we showed that these assumptions hold for our case-study and should extend to many binary ECSMs. Such shortcut formulas are extremely relevant in instances where the target algorithms exhibit numerous attack paths, as is the case for ECC implementations. In particular, if we consider powerful and complex attacks that target various vulnerabilities of ECC implementations jointly, then shortcuts such as presented in Chapter 3 become even more interesting in a time-constrained evaluation context.

A possible future direction is to extend the shortcut formulas we

propose in this thesis to windowed ECSMs, which process multiple bits of the secret scalar at the same time. In addition, one could investigate how to take into account both nearly worst-case horizontal attacks and lattice reduction attacks, by combining horizontal attack shortcuts and estimations for lattice attacks, such as provided by Goudarzi, Rivain and Vergnaud [74].

6.2 On the Worst-Case Security of ECC Point Randomization

In Chapter 3, we analyzed known input point horizontal attacks applied to ECSM implementations. Since point randomization is a natural countermeasure against these attacks, we analyzed its worst-case security in Chapter 4. We first identified two attack paths and in particular showed how to apply belief-propagation-based attacks [163] to a common example of point randomization. Following, we compared these costly attacks to simpler D&C attacks using the LRPM [78] and showed that the gap between the two is limited in this ephemeral secret setting. This results in a considerable speed-up of evaluations for the point randomization countermeasure. In a practical experiment, we considered a non-optimized point randomization procedure and an optimized one [87], and arrived at the conclusion that even simple efficiency-driven optimizations can lead to significant security improvements in our ephemeral secret/single-trace context. We then compared different coordinate systems, with respect to the worst-case resistance of the point randomization, and showed that this decision depends on the SNR and other implementation details of the ECSM. Finally, we extrapolated our results to harder-to-analyze 32-bit targets, and showed that high security can be achieved in software for the point randomization countermeasure.

Regarding future work, while we studied different implementation options of the point randomization, there are many options to implement ECC systems that remain to be analyzed. For instance, in case of ECDSA, if the randomized point is doubled on the fly and not precomputed in a classic Montgomery ladder (as the most significant bit of the secret scalar is equal to 1), then this additional leakage that does not depend on the secret scalar bit could potentially be exploited to recover the randomized point. Besides, different multiplication algorithms have been described in the literature. We focused on multiplications that share the same abstract architecture as a classic schoolbook multiplication, but other long integer multiplication algorithms such as the Karatsuba algorithm [95] or modular multiplication methods such as the

Montgomery multiplication [125] remain to be studied.

Furthermore, and similarly to our analysis of the cost vs. security trade-off between masking, shuffling and their combination, one possible direction is to compare and investigate any possible trade-off or combination of common ECC countermeasures, including but not limited to: point randomization, scalar blinding [43], exponent splitting [157] and Residue Number System (RNS) [62, 131] (of which variations were recently suggested to protect lattice-based Post-Quantum Cryptography (PQC) candidates [82]).

6.3 Amplified Bitslice Masking and Shuffling

In Chapter 5, we tackled the two most popular countermeasures to harden block cipher software implementations, namely masking [33] and shuffling [83]. We analyze their combination and study their cost vs. security trade-off. First, we improve D&C attacks targeting shuffling and show that, similarly to the point randomization's ephemeral secret in Chapter 4, recovering the shuffling permutation does not necessarily require complex attacks, and in particular for low SNR levels D&C attacks are nearly optimal. Our main contribution consists in suggesting a new masking + shuffling combination, which consists in shuffling shares of multiple values instead of tuples of shares. This combination results in exponentially increasing the impact of shuffling (which is a function of the number of shuffled values or operations) in the masking order. We show the relevance of our approach in the very popular bitslice implementation setting [21]. Finally, we discuss the limitations of algorithmic countermeasures, such as shuffling, on low cost software platforms with limited noise, and in practice we show that it is trivial to recover the full shuffling permutation with a simple profiled D&C attack.

An alternative research topic is to examine how the masking + shuffling configurations we propose in this thesis affect resistance against horizontal attacks. In addition, analyzing trade-offs between the granularity of the shuffling (i.e. the number of operations that are always performed sequentially) and refreshings (which were suggested as a countermeasure against horizontal attacks [13, 32]) appears as an interesting investigation.

One general conclusion of this thesis, is that despite worst-case attacks being theoretically preferable to ad hoc D&C or E&P strategies, depending on the target algorithm's structure and the device's leakage characteristics, simple exploitation approaches can come very close to

more complex worst-case approaches. This is shown in this thesis for SASCA against ephemeral secrets in the SPA context, e.g., the secret involved in the point randomization countermeasure analyzed in Chapter 4 or in the shuffling countermeasure analyzed in Chapter 5. However, in the DPA context, for example, against unprotected or protected AES implementations, we observe in practice in many works that the improvement factors (which are cipher dependent) that SASCA brings are far from negligible [163, 17, 29, 27]. For instance, Bronchain and Standaert use a different modeling of high-order masked implementation's factor graph to exploit a large number of useful intermediates very efficiently [29]. SASCA is also a very important evaluation tool for leakage-resilient primitives, where minimal differences in the SNR can lead to very large discrepancies in their security analysis [17].

From the previous observations, it is clear that an important future research direction is to point out cases where analytical attacks do not bring any significant improvements, in order to save time, effort and resources spent on worst-case evaluations. Equally as important, is to pinpoint the adverse cases where analytical attacks are necessary for high assurance/worst-case security assessments. This future work is additionally motivated by the increasing number of applications for embedded cryptography, products, devices, and even cryptographic schemes. With currently ongoing NIST standardization initiatives (PQC [127] and Lightweight cryptography [126]), it is clear that we need to target more efficient and systematic evaluations. In particular, most of the remaining PQC candidates [127] have many different building blocks and non-regular structures (as opposed to e.g., iterative and recursive ECSMs or repeated round function block ciphers). Investigating the worst-case security and the efficient evaluation of these schemes remain important research topics

Our research further showcases how shortcuts (e.g., based on shortcut formulas, the MI or model-based like the LRPM) can be very helpful in efficiently assessing the gap between attack strategies, predicting the outcome of tedious attacks and estimating the impact of countermeasures. For instance, our work in Chapter 5, suggests that MI shortcuts or the LRPM can be used as efficient preliminary tools to analyze ciphers and countermeasures. They offer a simple way to combine, examine and study the impact of different combinations of countermeasures, and even exhibit non-trivial effects, as shown in Section 5.3. An interesting study would be to apply these techniques to new schemes, with different and uninvestigated algorithms and operations. The goal is to gain more insight into the most efficient and impactful approaches to protect them against side-channel attackers and to study how shortcut methodologies

can support early decision making regarding side-channel countermeasures, similarly to our analysis of the point randomization in Chapter 4, and the masking + shuffling combination in Chapter 5. Another promising direction is to investigate if observations made in the more heuristic LRPM, for instance regarding the combination of masking and shuffling or the operations/intermediates usefulness in the case of ISW multiplications [32] or field multiplications in Chapter 4, can have counterpart proofs in more formal models such as the RPM [52].

Additional Contributions

Chapter A

Key Enumeration from the Adversarial Viewpoint

Key enumeration and key rank estimation are important parts of the security evaluation of cryptographic implementations. These methods allow post-processing the side-channel attack outcomes and determine the computational security of an implementation with respect to a full key recovery. In this work, we tackle a different question than key enumeration or rank estimation, which lies somewhere in between. In a realistic attack scenario, after an adversary has performed a D&C side-channel information extraction using a set of observations, she obtains a vector of probabilities or scores for each subkey, and the end goal is to perform a full key recovery. In this context, if all the subkeys have not been fully recovered immediately from the information extracted, the natural next step is key enumeration, up to a certain limit defined by her computational capabilities.

Depending on the attacks, the cryptographic operation(s) targeted, and the size of the key, enumeration can become the most time- and resource-consuming step. Hence, it is very useful (for an adversary) to efficiently estimate the key rank, and therefore to know whether it is worth to start enumerating the key candidates or if the collection of more side-channel observations is needed. We believe that this is an important problem from an adversarial point-of-view, that has not yet been investigated. We show in this work a simple solution for this purpose and propose a key-less or key-agnostic heuristic (i.e. that doesn't require the knowledge of the key) to efficiently approximate the rank of a key given the result of a side-channel attack. Our solution is based on the key rank estimation method from Glowacz et al. [68] using histogram convolutions, which we combine with information theoretic metrics to predict the rank of the full key from one single attack. Additionally, we

compare an adaptation of Choudary and Popescu's key-less guessing entropy estimation [35], which was proposed at CHES 2017, in this single attack setting to our method.

The rest of this chapter is organized as follows. First, Section A.1 introduces the required background and in particular Subsection A.1.3 describes more precisely the problem under investigation. Then, Section A.2 discusses our proposed method to estimate the rank without the knowledge of the key from an attack perspective. Section A.3 suggests an adaptation of the CHES 2017 proposal for rank estimation to the single-attack/attacker scenario. The results of the different methods are further shown in Section A.4. We finally discuss the usability and some limitations of our method and any key-agnostic method in Section A.5 .

A.1 Preliminaries and background

A.1.1 Key enumeration and rank estimation

In the following, we provide a brief overview of key enumeration and rank estimation proposals.

Key enumeration is an adversarial tool that allows testing key candidates without the knowledge of the key by listing the key candidates starting with the most likely one according to the attack results. For example, Veyrat-Charvillon et al. presented a deterministic algorithm for key enumeration that allows the optimal enumeration of full keys by decreasing order of probabilities, by reformulating the key enumeration problem as a geometric problem [161]. Ye, Eisenbarth and Martin proposed a so-called key space finding algorithm, that returns the enumeration workload for a given success probability, by considering the number of optimal guesses for each subkey [170]. Bogdanov et al. proposed a score-based key enumeration algorithm that reduces the high computational and memory complexity of previous proposals [23]. Martin et al. provided a new method by casting the key enumeration as an integer knapsack problem [114].

On the other hand, **rank estimation** is a part of the side-channel evaluator's tool set: given lists of discrete probability distributions for independent parts of the key and the correct key, a key rank estimation algorithm provides tight bounds for the position of the correct key among all possible ones (i.e. the number of guesses required to find the key following an optimal enumeration). In 2013, Veyrat-Charvillon, Gérard and Standaert showed that in the context of security evaluations

(for which the key is usually known), it is possible to estimate the rank of a key beyond the evaluator's practical enumeration capabilities [162]. Then, Bernstein, Lange, and van Vredendaal improved this rank estimation proposal and introduced a new method using polynomial multiplication to calculate lower and upper bounds for the ranks of all keys, by re-writing the problem of counting probabilities larger than the key's probability as finding the number of terms in a generalized polynomial satisfying a specific condition [19]. In parallel, Glowacz et al. proposed a new tool for rank estimation that is conceptually simpler and very efficient, based on histogram convolutions [68], which was later extended to key enumeration by Poussier, Standaert and Grosso [133]. At CHES 2017, a faster but heuristic approach was proposed by Choudary and Popescu [35]. They suggested to bound the key guessing entropy with information theoretic measures and inequalities, that do not require the knowledge of the correct key, estimated across multiple side-channel attacks.

A.1.2 Entropy, rank and guessing entropy

In the following we recall some definitions of IT and side-channel metrics that we use in the remainder of this chapter.

Entropy. The Shannon entropy H of a discrete random variable $\mathrm{X} \in \mathcal{X}$ following a probability distribution Pr is defined as:

$$\mathrm{H}(\mathrm{X}) = -\sum_{x \in \mathcal{X}} \Pr[x] \cdot \log \Pr[x]$$

Rank. The rank R, after a side-channel attack using a set $\mathcal{X}$ of inputs (e.g., plaintexts) and a corresponding set of leakages $\mathcal{L}$, provides the position of the correct key k in the sorted vector of $|\mathcal{K}| = 2^n$ (with n the size of k in bits) key candidate probabilities $\mathbf{p} = [p_1, p_2, ..., p_{|\mathcal{K}|}]$ in decreasing order, i.e:

$$\mathsf{Rank}(k) = i \text{ if } \Pr[k|\mathcal{X}, \mathcal{L}] = p_i$$

Guessing entropy. The guessing entropy GE [154] measures the average number of key candidates to test after a side-channel attack. It corresponds to the average rank of the correct key and is defined as:

$$\mathsf{GE} = \mathop{\mathbb{E}}_{k \in \mathcal{K}}[\mathsf{Rank}(k)] \tag{A.1}$$

A.1.3 Problem statement

Key enumeration algorithms offer a trade-off between the required number of side-channel observations and the computational power of the adversary. The use of such algorithms allows recovering the full key with fewer traces, often referred to as the data complexity of a side-channel attack, at the cost of computational/time complexity. However, in practice, it is not trivial for an adversary to know when to start the enumeration. Our goal is to tackle this problem by answering the following question: *Can the attacker infer if more side-channel measurements are required or if she expects a successful key recovery after enumeration, given the current attack outcome?*

To highlight the importance of this question, we picture the strategy that an adversary follows to recover the full key. When performing a side-channel attack it is common to follow a D&C strategy, in which parts of the key are targeted and possibly recovered independently. The target of a side-channel attack is an n-bit key $k \in \mathcal{K}$, divided into $N_p = \frac{n}{b}$ parts of b bits, called subkeys and denoted as k_i, for $1 \leq i \leq N_p$. A side-channel attack makes use of a set of q inputs $\mathcal{X}_q$ and the corresponding set of q leakages $\mathcal{L}_q$ (e.g., when targeting the AES Sbox output, the attacker observes, given an input x_i^j $(0 \leq j < q)$, the leakage l_i^j of $\mathsf{S}(x_i^j \oplus k_i)$). After the attack, the adversary obtains N_p lists of probabilities $P_i[k_i^*] = \Pr[k_i^* | \mathcal{X}_q, \mathcal{L}_q]$ where k_i^* refers to a subkey possibility out of the 2^b candidates. Attacks using Gaussian templates [34] or a linear-regression model [149] output probabilities, but for other distinguishers such as DPA [100] and CPA [25], a Bayesian heuristic extension is possible [161].

Let's assume that the adversary is able to perform key enumeration with respect to some computational effort e (e.g., $1 < e < 2^{50}$). A first attack strategy is shown in Algorithm 20. In that case, the attacker first collects a set of measurements, performs the attack, enumerates up to the first e most probable key candidates or until the correct key has been recovered. If the key has not been recovered, she then collects additional side-channel observations and repeats the process.

Algorithm 20 Greedy attacker's strategy.

Input. Enumeration effort e.

1: Collect a set of side-channel measurements.
2: Update the N_p probability lists P_i for each subkey k_i.
3: Enumerate up to e key candidates or until the correct key k is found.
4: **if** k not found **then**
5: Collect more side-channel measurements and go to step 2.

The greedy attacker strategy has one main drawback. Indeed, the attacker does not know if the key is reachable via enumeration after step 2. That is, she has no way to know how many times she has to loop over steps 2 to 6. More specifically, if (e.g.) w repetitions of side-channel measurements are required for the attack, the adversary spends an effort of $w \times e$ in enumeration. As the time complexity of such a method is high, it mitigates the original goal of enumeration, which is to trade a lower measurement complexity for a higher computational power.

The aforementioned issue could be avoided if the adversary could assess if the key is reachable with an enumeration effort at most e, using the currently available measurements. Following this idea, we now assume that the attacker is provided with a tool that, from the probability lists P_i and without the knowledge of the actual key, returns an approximate $\hat{\mathrm{R}}$ of the actual rank R. Using this tool, a more efficient attack strategy is shown in Algorithm 21. With this method, enumeration is executed only if the approximated rank $\hat{\mathrm{R}}$ is found to be within the enumeration effort e. If $\hat{\mathrm{R}}$ is close enough to R, this method is obviously more optimal than the previous one and achieves the desired trade-off between side-channel observations and computational power.

Algorithm 21 Efficient attacker's strategy.

Input. Enumeration effort e and a key-less rank estimation.

1: Collect a set of side-channel measurements.
2: Update the N_p probability lists P_i for each subkey k_i.
3: From $(P_i)_{1 \leq i \leq N_p}$ compute an approximation $\hat{\mathrm{R}}$ of the rank R.
4: **if** $\hat{\mathrm{R}} \leq e$ **then**
5: Enumerate up to e key candidates or until k is found.
6: **if** k not found **then**
7: Collect more side-channel measurements and go to step 2.
8: **else**
9: Collect more side-channel measurements and go to step 2.

It is worth mentioning that, as a side advantage, the existence of such a tool would also give information to the adversary on whether or not the attack will succeed eventually. Indeed, observing the trend of $\hat{\mathrm{R}}$ either gives some confidence in the efficiency of the attack if it decreases steadily as the number of measurements increases, or shows that the attack has little chance to eventually recover the key (or a significant part of it) otherwise.

A.1.4 Histogram-based rank estimation

Among the different proposals for rank estimation, we use the histogram-based approach of Glowacz et al. [68]. This algorithm provides efficiently tight bounds for the rank of the key and gives the probability distribution of the full key expressed as a histogram. Given the N_p lists of the log probabilities of the subkeys $LP_i = \log(\Pr[k_i^*|\mathcal{X}_q, \mathcal{L}_q])_{k_i^*}$, the method of Glowacz et al. starts by constructing the corresponding N_p histograms $H_i = \text{hist}(LP_i, \text{bins})$ where bins is a set of N_{bin} equally-sized bins, used for all the histograms. The convolution of two histograms is denoted as $\mathsf{conv}(H_i, H_j)$. Knowing the key k, its log probability is:

$$\log(\Pr[k|\mathcal{X}_q, \mathcal{L}_q]) = \sum_{i=1}^{N_p} \log(\Pr[k_i|\mathcal{X}_q, \mathcal{L}_q])$$

The following steps are described by Algorithm 22. In a nutshell, the rank estimation algorithm provides a very efficient way to estimate and bound the number of key candidates with a probability higher than k.

Algorithm 22 Rank estimation.

Input: The key log probability $\log(\Pr[k|\mathcal{X}_q, \mathcal{L}_q])$ and the histograms $(H_i)_{1 \le i \le N_p}$.
Output: An estimation of k's rank and corresponding bounds.

initialization: $H_{\text{curr}} = H_1$
histograms convolution:
for $i = 2$ to N_p **do**
    $H_{\text{curr}} = \mathsf{conv}(H_{\text{curr}}, H_i)$
rank estimation:

$$\text{R} \approx \sum_{i=\mathsf{bin}(\log(\Pr[k|\mathcal{X}_q,\mathcal{L}_q]))}^{N_p \cdot N_{\text{bin}} - (N_p - 1)} H_{\text{curr}}(i).$$

The function bin returns the bin index containing the given log probability. The estimated rank is then lower (lb) and upper (ub) bounded by tracking the quantization error of the histograms as:

$$\text{R}_{\text{lb}} = \sum_{i=\mathsf{bin}(\log(\Pr[k|\mathcal{X}_q,\mathcal{L}_q]))+N_p}^{N_p \cdot N_{\text{bin}} - (N_p - 1)} H_{\text{curr}}(i),$$

and:

$$\mathrm{R}_{\mathrm{ub}} = \sum_{i=\mathsf{bin}(\log(\Pr[k|\mathcal{X}_q,\mathcal{L}_q]))-N_p}^{N_p \cdot N_{\mathrm{bin}}-(N_p-1)} H_{\mathrm{curr}}(i),$$

By increasing the number of bins (or decreasing the size of the bins), the tightness of the bounds can be arbitrarily reduced. In the rest of this chapter, and for the practical experiments, we use a number of bins large enough such that the tightness is below 1 bit. As this precision is enough for our experiments, we consider the estimated rank as the "true" rank and ignore the bounds. We emphasize that the histogram convolutions do not require the knowledge of the key, and only the rank estimation itself does. For the rest of this chapter, we use the final histogram constructed, which corresponds to the full key distribution.

A.2 Entropy-based key-less rank approximation

The entropy of the key candidate probabilities produced by an attack intuitively brings some information on the outcome of the attack, since it measures the uncertainty on the key, which amounts to the number of bits of information left to recover on the key. For instance, an attack ran on data that is uncorrelated with the key would tend to attribute average probabilities to most candidates, yielding a high entropy (n bits in the extreme case where all candidates have a probability of 2^{-n}, while the average rank is 2^{n-1}). On the other hand, an attack performing extremely well would give a high probability to the correct key candidate and very low ones to all other candidates. In that case, the entropy tends to 0 (like the log rank) as the probability of the wrong candidates also tends to 0. This intuition that in realistic cases entropy and rank are linked – although it's a loose link, as one can show that entropy and rank can diverge in specific cases – leads us to consider the use of the entropy to estimate the remaining enumeration effort of an attack.

In the following, we present how the entropy of the full key can be estimated using the histogram obtained with the convolution-based method of Glowacz et al [68]. Although the histogram is only a compressed representation of the full key probability distribution, it still is an excellent tool to analyze it. To estimate the entropy, we require a proper probability distribution that sums up to 1, hence it is required to normalize the key distribution. This is done by ensuring that the sum of the exponential of log probabilities given by the bins of all keys in all histograms sum to 1. Furthermore, it is highly recommended to normalize the distribution of the subkeys, prior to the histogram convolution, to avoid that the estimated metrics are weighted by the distributions of

the subkeys. The entropy of the full key after a side-channel attack is estimated using the corresponding histogram (H,bin) as:

$$\begin{aligned} \mathrm{H} &= -\sum_{k^* \in \mathcal{K}} \Pr[k^*] \cdot \log \Pr[k^*] \approx \sum_{k^* \in \mathcal{K}} \exp(\mathsf{bin}(k^*)) \cdot \mathsf{bin}(k^*) \\ &\approx \sum_{i=1}^{N_p \cdot N_{\mathrm{bin}} - (N_p - 1)} H(i) \cdot \exp(\mathsf{bin}(i)) \cdot \mathsf{bin}(i) \end{aligned}$$

To simplify the above equation, here $\mathsf{bin}(k^*)$ provides the bin index of the log probability of k^*. Note that while the original sum is over all 2^n key candidates, which is impossible to perform, the entropy estimation using the histogram (which is a compression of the full key distribution) only requires to sum over all N_{bin} bins. We denote by $\tilde{\mathrm{H}}$ the estimation of the entropy using the histogram. Note that bounds on this estimation are based on the quantization error of the histogram similarly to the rank and are given by:

$$\mathrm{H}_{\mathrm{lb}} = \sum_{i=1}^{N_p \cdot N_{\mathrm{bin}} - (N_p - 1)} H(i). \exp(\mathsf{bin}(i - N_p)).\mathsf{bin}(i - N_p)$$

$$\mathrm{H}_{\mathrm{ub}} = \sum_{i=1}^{N_p \cdot N_{\mathrm{bin}} - (N_p - 1)} H(i). \exp(\mathsf{bin}(i + N_p)).\mathsf{bin}(i + N_p)$$

A.3 Adaptation of the CHES 2017 key-less GE

At CHES 2017, Choudary and Popescu [35] consider a number of information theoretic measures and inequalities that are easy and efficient to estimate in order to bound the guessing entropy of the key. However, in their proposal, they make a distinction between a key-agnostic guessing entropy and one that requires the knowledge of the key. The framework and the definitions in [35] are actually inaccurate in this respect as they suggest that the GE is the *actual* key rank while it is the *average* key rank. The keyed and key-less versions are equivalent in case the templates used in the key-less estimation are perfect, so the difference between both definitions only lies in the knowledge of the key. This key-less guessing entropy, that we denote by GE_{kl}, is computed given the sorted vector of the $|\mathcal{K}| = 2^n$ key candidate probabilities $\mathbf{p} = [p_1, p_2, ..., p_{|\mathcal{K}|}]$ obtained after a side-channel attack as:

$$\mathsf{GE}_{kl} = \sum_{i=1}^{|\mathcal{K}|} i \cdot p_i \tag{A.2}$$

This definition is clearly different from the one we use in side-channel analysis defined by Equation A.1. While Equation A.1 requires the knowledge of the correct key, the keyless GE given by Equation A.2 does not use any knowledge of the key. The metrics are different and this can be observed in the work of Choudary and Popescu [35] (e.g., Figures 2 and 3).

Choudary and Popescu notice in their work that GE_{kl} is impossible to compute since the sum is over all full key candidates. For that purpose they use common measures and bounds described in information theory literature. Since this work provides a key-less rank estimation tool, a natural alternative to our previous proposal is to try to adapt this key-agnostic GE to our single-attack context. This alternative is again only heuristic since the bounds are only valid on average as pointed out in [35]. To describe this alternative solution, we use again the histogram-based PDF estimation provided by Glowacz et al.'s work. Here, GE_{kl} corresponds to the sum of the position of every key candidate weighted by its probability and it can be estimated as:

$$\mathsf{GE}_{kl} = \sum_{i=1}^{|\mathcal{K}|} i \cdot p_i \approx \sum_{k^* \in \mathcal{K}} \left(\sum_{j=\mathsf{bin}(k^*)}^{N_p \cdot N_{\mathrm{bin}} - (N_p - 1)} H(j) \right) \cdot \Pr[k^*]$$

Then, we sum across all keys and all bins in the histogram, yielding the estimation of the key-less guessing entropy $\tilde{\mathsf{GE}}_{kl}$:

$$\tilde{\mathsf{GE}}_{kl} \approx \sum_{i=1}^{N_p \cdot N_{\mathrm{bin}} - (N_p - 1)} \left(\sum_{j=i}^{N_p \cdot N_{\mathrm{bin}} - (N_p - 1)} H(j) \right) \cdot \exp(\mathsf{bin}(i))$$

The bounds on this histogram-based approximation are given by:

$$\mathsf{GE}_{kl,\mathrm{lb}} = \sum_{i=1}^{N_p \cdot N_{\mathrm{bin}} - (N_p - 1)} \left(\sum_{j=i+N_p}^{N_p \cdot N_{\mathrm{bin}} - (N_p - 1)} H(j) \right) . \exp(\mathsf{bin}(i - N_p))$$

$$\mathsf{GE}_{kl,\mathrm{ub}} = \sum_{i=1}^{N_p \cdot N_{\mathrm{bin}} - (N_p - 1)} \left(\sum_{j=i-N_p}^{N_p \cdot N_{\mathrm{bin}} - (N_p - 1)} H(j) \right) . \exp(\mathsf{bin}(i + N_p))$$

The previous bounds on the entropy and the key-less guessing entropy are not directly relevant to our work. Precisely, the ones in [68] require the knowledge of the key (i.e. an evaluation setting) while we aim at key-less rank estimation, and the ones in [35] only become tight

on average over many experiments (i.e. also in an evaluation setting) while we aim to estimate the rank of single experiments on-the-fly. For the same reason, bounds on the entropy are also irrelevant in our adversarial setting. Subsequently, the rest of this work ignores the bounds and only focus on the heuristic evaluation of these metrics.

A.4 Simulated and real experiments

In the following sections, we investigate whether the previous metrics can be used to approximate the rank of the correct key after a single side-channel attack. For this purpose, we evaluate two average absolute differences (for multiple side-channel attacks): the first between the logarithm of the rank log(R) and the estimated entropy $\tilde{\mathsf{H}}$, and the second between log(R) and the logarithm of $\tilde{\mathsf{GE}}_{kl}$. We use both simulated and real experiments. We start by describing our experimental setups and then show practical results.

A.4.1 Experimental setups

Simulated leakages: We simulated side-channel leakages of the 16 Sbox outputs of the first round of an unprotected AES. For each byte x_i out of the 16, we model the leakage of x_i as $\mathsf{HW}(x_i) + b$, where b represents an independent noise distributed according to a normal distribution with mean 0 and standard deviation 10.

Real leakages: We target a custom constant-time C implementation of an unprotected AES with T-tables. The code runs on an ARM Cortex-M3 microcontroller running at 83 Mhz, mounted on an Arduino Due board. We acquired EM measurements with a Langer near field RF-U 5-2 probe and a Lecroy 610Zi oscilloscope at a sampling rate of 1 GHz. We synchronized the traces at the beginning of the AES computation using a trigger signal. Similarly to the simulated case, we target the output of the Sboxes of the first round. For each of the 16 target bytes, we selected the POI that exhibits the highest correlation with the corresponding Sbox output value, using a first set of 10,000 traces with a known key.

We performed a Template attack [34] for both simulated and real leakage experiments. For the real experiments, we use a set of 100,000 traces with known key and plaintext for the template building phase.

A.4.2 Evaluation results

We show here the practical results using the estimation of the entropy $\tilde{\mathsf{H}}$ and the estimation of the key-less guessing entropy $\tilde{\mathsf{GE}}_{kl}$ to approximate the rank of the key R after a side-channel attack. We additionally compare the performance of both metrics. We recall that we estimate the rank using the histogram method of Glowacz et al. and that we used a large enough number of bins so that the bounds are tight enough for us to use the estimated rank and ignore the bounds. In the following, we refer to the approximation of log(R) using the entropy as the $\tilde{\mathsf{H}}$-based approximation and the one using the logarithm of the key-less guessing entropy as the $\tilde{\mathsf{GE}}_{kl}$-based approximation.

As a preliminary experiment, we ran a single attack against both simulated and real traces. We incremented the number of attack traces until the rank reached one. The results are depicted in Figure A.1. The left (resp. right) part of the figure shows the results for simulated (resp. real) traces. In each case, the x-axis represents the number of attack traces, and the y-axis is used to represent the different metrics in $\log_2$ scale. We notice that the entropy-based approximation is closer to the logarithm of the rank than the one based on the key-less guessing entropy. More specifically, for the simulated traces the $\tilde{\mathsf{H}}$-based approximation remains within less than 10 bits of log(R), while the gap with the $\tilde{\mathsf{GE}}_{kl}$ goes up to 25 bits. The real experiments show less optimistic results. Indeed, while $\tilde{\mathsf{H}}$ remains mainly within 5 bits of log(R), we can observe a fairly large gap of 30 bits when the rank is around 2^{40}. Results are worse for $\log(\tilde{\mathsf{GE}}_{kl})$ with a maximal gap of 50 bits.

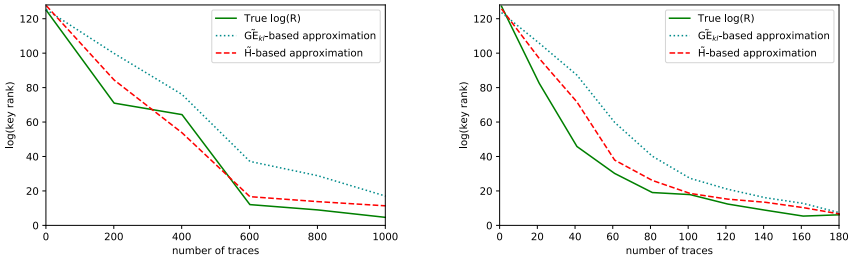


Figure A.1: Comparison of $\log(\tilde{\mathsf{GE}}_{kl})$, $\tilde{\mathsf{H}}$ and log(R) for a single side-channel attack on the AES Sbox output. Simulated traces (left) and real EM traces (right).

The previous experiment showed some results for a single attack and that the entropy is neither an upper bound or a lower bound on the logarithm of the rank. While this gives some insight about the interest of

the considered metrics, it does not allow deriving any general conclusion.

Next, we examine how the $\tilde{\mathrm{H}}$-based approximation and the $\tilde{\mathsf{GE}}_{kl}$-based one perform on average over many independent experiments. For that purpose, we evaluate the average absolute difference in bits, first between $\tilde{\mathrm{H}}$ and log(R), then between log($\tilde{\mathsf{GE}}_{kl}$) and log(R). This is performed over 100 independent attacks for the simulated and the real traces. The results on the simulated traces are shown in Figure A.2. The x-axis corresponds to log(R). The y-axis on the left (resp. right) part of the figure gives the mean (resp. standard deviation) of the distance between each measure and the rank. For the left part of the figure, we additionally plot the maximal distance obtained over the 100 attacks in corresponding dashed curves. We notice that the entropy clearly outperforms the key-less guessing entropy when approximating the rank for a single attack. First focusing on the average distance on the left part of the figure, the entropy stays within less than 10 bits of R. The maximal difference we obtained among the 100 experiments is slightly above 20 bits. However, the mean distance of log(R) to log($\tilde{\mathsf{GE}}_{kl}$) is in most cases above 10 bits and below 25 bits, with a maximum above 45 bits. Also considering the standard deviations of these distances on the right part of the figure, it is reasonable to consider that the entropy provides a more reliable estimation of R than the key-less guessing entropy, since the standard deviation of the distance to $\tilde{\mathrm{H}}$ is lower than the one to log($\tilde{\mathsf{GE}}_{kl}$), the first one is less likely to deviate from its mean value which is below 10 bits.

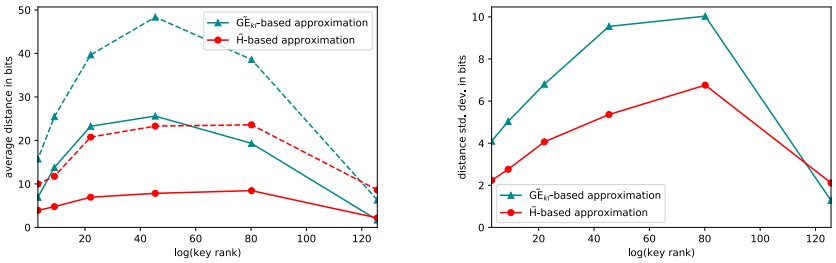


Figure A.2: Simulated HW leakages: (Left) The average distance from the rank to both the $\tilde{\mathrm{H}}$-based approximation and the $\tilde{\mathsf{GE}}_{kl}$-based approximation as function of the logarithm of the rank and the maximal distance observed in corresponding dashed lines. (Right) the distances' standard deviations.

The results for real traces are shown in Figure A.3. Again, the x-axis represents log(R) and the y-axis on the left (resp. right) part of the

figure represents the mean (resp. standard deviation) of the distances, computed over 100 independent attacks. The experiments on real traces coincide with the simulated ones. Accordingly, the entropy-based approximation provides a better estimate of log(R) than the approximation based on the key-less guessing entropy. The average distance between log(R) and $\tilde{\mathrm{H}}$ is below 12 bits while the average distance to log($\tilde{\mathsf{GE}}_{kl}$) is always above the average distance to $\tilde{\mathrm{H}}$. The right part of Figure A.3 is similar to the right part of Figure A.2, as the standard deviation of the distance to the entropy is lower than the one to the key-less guessing entropy, confirming that for a real side-channel attack, $\tilde{\mathrm{H}}$ provides a better approximation of log(R) than log($\tilde{\mathsf{GE}}_{kl}$).

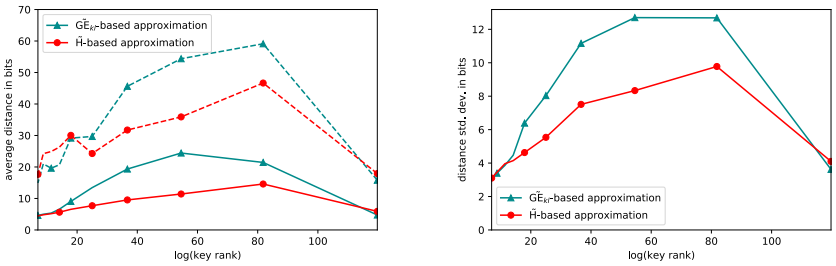


Figure A.3: EM traces: (Left) The average distance from the rank to both the $\tilde{\mathrm{H}}$-based approximation and the $\tilde{\mathsf{GE}}_{kl}$-based approximation as function of the logarithm of the rank and the maximal distance observed in corresponding dashed lines. (Right) the distances' standard deviations.

We highlight the fact that all predictions have a higher variance for middle ranks, which are the most interesting for both evaluators and attackers, as it is typically the range of ranks where enumeration turns from being unfeasible to practically feasible. This has been previously observed by Martin et al. in an evaluation setup [114] with the possibility to perform multiple attacks with the knowledge of the key. Our results and the ones of Martin et al. show that the interesting ranks are the hardest to estimate for an evaluator, and especially in the context of a real attack with the purpose of recovering the key.

Our proposed metric cannot mathematically approximate nor bound the true rank of the key after a single side-channel attack. However, we experimentally show that the entropy tends to stay within reasonable limits from the logarithm of the rank (provided that the attack does not suffer from errors (e.g., due to a wrong model assumption). As a result,

we believe it can be used in a more efficient strategy by trading data complexity for computational effort as illustrated by Algorithm 21 by enumerating key candidates up to (or slightly above) $2^{\tilde{\mathrm{H}}}$.

A.5 Discussion and limitations

Any attempt to predict the rank from one single attack without the knowledge of the key suffers from a specific caveat. It is possible that the attack is not carried out correctly and is converging towards a wrong key (due for instance to wrong intermediates, wrong assumptions about the leakage or unknown countermeasures). The entropy and the key-less guessing entropy would then decrease as the attack tends towards the wrong key candidate, while the rank of the correct key would not. This behavior does not only affect the entropy and the key-less guessing entropy but most probably any metric estimated without the knowledge of the correct key.

Another aspect to consider that affects the considered metrics is the key size. This can be pictured through a simple example: let's consider two attacks that both aim at recovering a bit b whose value is 1, and output two probability distributions $\mathbf{p_1} = [\Pr_1[b = 0] = 0\,, \Pr_1[b = 1] = 1]$ and $\mathbf{p_2} = [\Pr_2[b = 0] = 0.45\,, \Pr_2[b = 1] = 0.55]$. Both attacks achieve a rank of one since the correct value of b has the highest probability. On the other hand, the entropy values are quite different. The entropy of the first attack is equal to 0, which is equal to the logarithm of the rank. For the second attack, the entropy of b is higher and equal to 0.99277, albeit the correct value of b is ranked first. These discrepancies can be observed for small keys, but vanish for larger key sizes. This illustrates how independent conclusions on subkeys can be quite misleading when trying to infer conclusions on full key recovery.

To demonstrate this effect, we performed the same experiments as described in the previous section. We estimated the average distance between log(R) and the entropy and then between log(R) and the key-less guessing entropy, but across different key sizes. For each key size, we performed 100 attacks. We normalized the distance with respect to the size of the key in bytes. Indeed, normalizing makes the distances comparable for different key sizes, and allows to infer conclusions based on the distance per byte. As an example, a minor distance for one key byte between R and its key-less prediction is critical, but not so relevant for the full key. The results are given in Figure A.4, for both the simulated traces on the left and real traces on the right. The dashed line indicates the maximum values observed. For the simulated experiments, it was

possible to perform experiments on large keys of up to 64 bytes, and up to the AES-128 key size for the real traces we measured. We used 400 attack traces for the simulations, leading to a rank of approximately 2^{55}, and 70 traces for the real attack to achieve a rank around 2^{30} for a 128-bit key (with proportional ranks for smaller key sizes). This was chosen to focus on the interesting ranks and we did not notice any considerable differences for other ranks, when it comes to the effect of the key size on the distance to log(R) of either the $\tilde{\mathrm{H}}$-based approximation or the $\tilde{\mathsf{GE}}_{kl}$-based one.

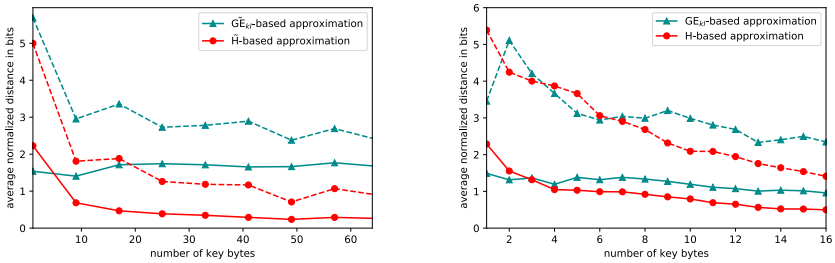


Figure A.4: Average distances between the log of the rank, the entropy and the log of the key-less guessing entropy as function of the number of key bytes. Maximal distances observed in corresponding dashed lines. Simulated traces on the left and real traces on the right.

We first observe that the normalized difference indeed decreases when the key size increases, confirming our intuition. Moreover, the trend starts to settle for both simulated and real traces once realistic full key sizes are reached. First, for a 1-byte key size, we can observe on average a two-bit difference between log(R) and $\tilde{\mathrm{H}}$ and a lower difference between log(R) and log($\tilde{\mathsf{GE}}_{kl}$). On the other hand, for a 16-byte key size, the distance between the rank and the entropy-based prediction drops to around 0.5 bits of error per byte for both the simulated and the real traces, while the distance between the rank and $\tilde{\mathsf{GE}}_{kl}$ seems to settle at an average distance of 1.5 bits of error per byte even for larger key sizes. For the maximal value, we observe the same trend as previous experiments. The distance to the key-less guessing entropy is higher than the one to the entropy in most cases.

Overall, two conclusions can be drawn from this experiment. First, it confirms that the entropy-based estimation seems to be a better tool to approximate the rank than the key-less guessing entropy once real key sizes are reached. Second, it shows that as expected, it is better to estimate the security level in an adversarial scenario on the full key than

on a small part of the key, such as a subkey.

A.6 Conclusion

In this chapter, we described a heuristic way to infer an approximation of the key rank for one single attack without the knowledge of the key. This corresponds to a realistic attack scenario, where the adversary aims at figuring out if the correct key can be reached through enumeration. Our proposal helps to devise an optimal attack strategy to trade data complexity for computational effort when possible. For that purpose, we showed that the remaining entropy of the full key can be estimated using the histogram built with the rank estimation method from Glowacz et al. without the knowledge of the key. We showed experimentally that the entropy of the full key distribution after a side-channel attack is close to the logarithm of the rank on both simulated data and real EM side-channel measurements of an AES implementation. We compared this entropy-based approximation of the rank, to a single-attack adaptation of the key-less rank estimation method of Choudary and Popescu [35]. We additionally discussed factors that may affect the accuracy of the entropy (and any measure that lacks knowledge of the key or its probability) as a predictor of the rank.

Further research might investigate if the behavior observed in this work is common to different side-channel datasets. Moreover, it would be interesting to investigate if the tool described in this work can identify possible wrong assumptions about the implementation or device that can hinder the success of the attack. Alternatively, an interesting direction is to propose a more precise key-agnostic technique or metric to approximate the rank of the correct key in the single attack scenario.

Chapter B

Blind Side-Channel SIFA

As opposed to SCA, which are *passive* attacks, Fault Injection (FI) attacks are *active* techniques that aim to inject errors into a target device, during the execution of a cryptographic function. These injections can be accomplished by several tampering means with varying granularity. Clock/voltage glitches can induce fairly coarse faults [101], while focused laser beams [159] can produce fine-grained errors. Typically, a fault is injected at a specific point in time and then algebraic approaches or statistical distinguishers are used to derive the secret key. The first fault attack was published by Boneh, DeMillo and Lipton in 1997 [24]. Expanding, Biham and Shamir introduced Differential Fault Attacks (DFA), collecting pairs of faulty and fault-free ciphertexts and using the combined knowledge of the fault induced and the difference between the ciphertexts to recover the key [22]. Very recently, Statistical Ineffective Fault Attacks (SIFA) have been proposed by Dobraunig et al. as very powerful key recovery strategies on symmetric cryptographic primitives' implementations [50]. Specifically, they have been shown to bypass many common countermeasures against faults such as duplication or n-plication [11], error-detecting codes [20] and infection [158], and to remain applicable even when side-channel countermeasures are deployed [49].

In this work, we investigate combined and profiled side-channel and fault attacks and show that a profiled SIFA-like attack can be applied despite not having any direct ciphertext knowledge. The proposed attack exploits the ciphertext's side-channel and fault characteristics to mount successful key recoveries, even in the presence of masking to protect the key and duplication countermeasures. This result is relevant for applications such as the session key derivation used in the EMV payment scheme [42] or when using any modes of operation that limit the

adversary's access to the plaintext and ciphertext. We stress that the attack relies on the fact that the ciphertext is not masked since it is an ephemeral secret. We discuss alternatives in Section B.5. The rest of this chapter is organized as follows, we offer background in Section B.1 and describe the new attack in Section B.2. The attack is analyzed with simulations in Section B.3 and is experimentally verified in Section B.4 on an ARM Cortex-M4 based Micro-controller.

B.1 Background

In this section, we first present a streamlined survey of fault attacks bypassing limited information on inputs/outputs or restricted access to the device, which are related to our work. Then we provide a brief description of SIFA.

B.1.1 Related works

The FI literature presents several attack strategies that adapt to various cases of limited knowledge or control over the plaintext and ciphertext. To deal with such restrictions, Fuhr et al. introduced Statistical Fault Attacks (SFA), that only require access to faulty ciphertexts [63]. Towards similar goals, Korkikian, Pelissier and Naccache designed a blind fault attack that does not require the knowledge of either plaintext or ciphertext and is applicable to any round [103]. However, it relies on the possibility to encrypt the same unknown plaintext multiple times, on the observation of the number of faulty ciphertexts and on specific fault models. Should the designer opt for detection-based countermeasures that limit access to faulty ciphertexts, Clavier introduced Ineffective Fault Attacks (IFA) that use ineffective faults (induced faults that do not change the internal state of the cipher) to probe the intermediate values of an algorithm [37].

In addition, it is worth mentioning that the restrictions imposed to the FI adversary can often be bypassed via the usage of side-channel analysis, leading to combined attacks. For instance, Roche, Lomné and Khalfallah present an SCA-assisted DFA that utilizes pairs of valid and faulty ciphertexts and relies on the ability to repeat plaintexts and to observe through side-channel the leakage of the unmasked faulty ciphertexts, while the later are manipulated by the fault detection mechanism [141]. This attack can be prevented if the ciphertexts remain masked during the fault detection procedure. Similarly, Saha et al. propose a technique that requires side-channel leakage of the final comparison but no knowledge of the faulty ciphertexts [142]. It is based on

the access to the bytewise HD between the correct and the faulty ciphertext for each fault injection and requires a precise multiple bit-reset or bit-set fault model. Following, they also present another profiled attack where ciphertexts are unknown and the adversary solely observes if a fault is detected or not [143]. Such a binary answer is particularly easy to obtain, simply by observing the high-level protocol which the cipher is a part of. However, it requires to repeat the same unknown plaintext during the fault profiling phase and the attack phase, and being able to exploit very precise fault effects.

The previously mentioned attacks are prevented if the fault detection is correctly masked or if plaintexts cannot be repeated. Some additionally require ideal side-channel information, while in practice side-channel leakages are noisy. We show in the remainder of this chapter that it is possible to bypass both fully masked fault detection mechanisms and the impossibility to repeat plaintexts, at the cost of both side-channel leakage and fault distribution profiling.

B.1.2 Statistical ineffective fault attacks

SIFA are a combination of SFA [63] and IFA [37] and were put forward by Dobraunig et al. [50]. SFA bypasses DFA's requirement for repeated plaintexts and IFA uses only ineffective faults to attack the intermediate values. Since ineffective faults cannot be detected by classic fault countermeasures, SIFA can exploit many practical implementations protected with common software countermeasures against faults, i.e. it only requires access to the ineffectively faulted ciphertexts.

In particular, the authors demonstrate that a fault injected at the input of the MixColumns step of the 9th round introduces a bias on the AES state, which is otherwise uniformly distributed. Knowing the ciphertexts, and by making a hypothesis on 4 bytes of the last round key (2^{32} guesses), it is possible to backtrack to the biased state. Incorrect key hypotheses result in a uniformly distributed state, while for the correct hypothesis the state distribution is biased. Such bias is easily detected using the squared euclidean imbalance distinguisher which measures the distance between the obtained hypothetical distributions and the uniform distribution. Moreover, SIFA is robust against noise introduced by failed fault inductions (for example in dummy rounds), but can be impeded if the probability of inducing an ineffective fault is negligible or if the resulting distribution of ineffective faults is uniform or close to uniform.

While the work of Dobraunig et al. [50] focuses on the penultimate round, they point out that SIFA can be performed on the last round as

well, only requiring a hypothesis on a single key byte. However, in this case the distribution of the state is far from uniform for all key byte guesses and thus the 10th-round SIFA must acquire information on the bias. If not known a priori, such information can be learned through a process of fault profiling, typically on an open device that is identical to the device under attack. SIFA is also applicable to masked implementations as demonstrated in [49], where it is shown that faulting only one share during the Sbox computation is enough to mount a successful attack.

B.2 Blind side-channel SIFA

This section describes the attack proposed in this work, named blind side-channel SIFA. The attack focuses on the last (10th) AES round and targets a single key byte, utilizing both SCA and FI. It can be straightforwardly adapted for different symmetric ciphers, as well as SFA techniques.

B.2.1 Attack context and motivation

This work focuses on a protected cipher implementation (AES-128) that deploys both higher-order masking against SCA and duplication-based fault detection against FI. We assume that the fault detection procedure is implemented in a protected manner and unmasking is performed solely on correct ciphertexts, i.e. the combined attack of Roche et al. [141] is no longer applicable. Furthermore, we consider an attacker that has no access or control over the plaintext and ciphertext, both are only assumed to be random. Thus, straightforward high-order DPA [33, 121] or SIFA are not applicable, since they both rely on plaintext or ciphertext knowledge. Likewise, the attacks of Korkikian et al. [103], Saha et al. [142, 143] are not applicable since the plaintext cannot be repeated. Similarly to Saha et al. [143] though, we assume that the attacker can observe the detection of a fault, either when a detection-based countermeasure is triggered within the AES implementation or through an error message in a higher-layer protocol.

Such a restricted attack scenario can impose severe limitations on the adversary, since it disables several effective tools from his arsenal. The main attack strategy that appears viable against such an implementation is a blind high-order SCA. A profiled version of this attack would follow the lines of Hanley, Tunstall and Marnane [80] or analytical/algebraic attacks [163], i.e. profiling and combining all shares of at least two intermediate values. An unprofiled version would follow the

lines of Linge, Dumas and Lambert-Lacroix [108], or Clavier, Reynaud and Wurcker [41], using the joint distribution of multiple intermediates' leakages. All high-order attacks however are impacted by the noise amplification and the exponential relation between attack traces and the masking order [33]. Intuitively, a blind attack alters the usual data complexity growing exponentially in the masking order d, to grow exponentially in $2 \cdot d$. Such a conundrum motivates us towards a combined attack that is not subject to the noise amplification, while still dealing with the unknown plaintext and ciphertext. However, our attack comes at the cost of both side-channel and fault profiling, which we describe in the next sections.

B.2.2 Working principle

Blind side-channel SIFA consists of two phases. The first phase requires access to an open training device supplied with random plaintexts, and knowledge of the ciphertexts and keys. Such a device is used to profile the side-channel leakage of the ciphertext (Section B.2.2.1) and subsequently profile the biased fault distribution of the ciphertext (Section B.2.2.2). Note that for the profiling phase we do not require disabling the fault detection countermeasure. The second phase (actual attack) is performed on a closed device where the ciphertext is unknown. The attacker injects faults while observing the side-channel leakage of the ciphertext resulting only from ineffective faults. Subsequently a key recovery strategy is deployed (Section B.2.2.3).

B.2.2.1 Side-channel profiling.

During the first phase, the attacker profiles the ciphertext leakage using side-channel measurements and a deterministic leakage function such as the HW. A ciphertext byte c's leakage l is modeled as: $l = \alpha \cdot \mathsf{HW}(c) + \beta + N$, where α and β are constant terms and $N \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise. Subsequently, the attacker uses pairs of known ciphertext bytes and corresponding leakages (c_i, l_i) to estimate the templates $(L|\mathsf{HW}(c) = h) \sim \mathcal{N}(\mu_h, \sigma^2)$. During the actual attack, these estimations will allow us to compute the likelihood $\Pr[\mathsf{HW}(c) = h|l] \propto \mathcal{N}(l|(\mu_h = \alpha \cdot h + \beta, \sigma^2))$ and recover a probability vector on $\mathsf{HW}(c)$. It is worth mentioning that this targeting of the ciphertext is based on the assumption that it is unmasked to be stored in memory or transferred to be used by other applications, e.g., as a derived ephemeral or session key. Additionally, the attack can also target the plaintext and the first round instead.

B.2.2.2 Ineffective fault profiling.

In this phase, the attacker must characterize the ineffective fault distribution of the ciphertext. To do so, he injects faults during the 10th round of AES in order to bias the Sbox output and observes the resulting correct ciphertexts. In the following, we use apostrophes to denote faulted values, e.g., a' is the biased version of a. In this round, we denote the true key byte as k, the biased Sbox output as s' and the corresponding ciphertext byte by c', i.e. when no fault is detected, $c' = k \oplus s'$ as depicted in Figure B.1. For brevity we omit the ShiftRows operation.

The faults can be injected before or during the masked SBox execution, since targeting non linear operations is a necessary requirement of SIFA on masked implementations [49]. The attacker acquires N_{fp} ciphertexts $(c'_i)_{0 \le i < N_{fp}}$ (filtered by fault detection) and uses the knowledge of the true key k to backtrack to $(s'_i)_{0 \le i < N_{fp}}$. Consequently, he computes the values $(\mathsf{HW}(s'_i \oplus k^*))_{0 \le i < N_{fp}}$ and builds the fault templates $\Pr[\mathsf{HW}(c')|K = k^*]$ for all candidates $k^* \in \mathbb{F}_{2^8}$, using histograms. We denote these fault templates by $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{255}$, which provide the probability $\Pr[h|k^*] = \mathcal{D}_{k^*}(h)$ of observing a ciphertext with HW h if the key byte is k^*.

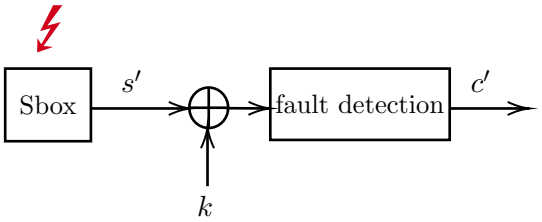


Figure B.1: Fault injection during the 10th round of AES.

B.2.2.3 Key recovery strategies.

During the second phase of the attack, the adversary has no access to or control over the ciphertexts, thus he is limited to injecting faults and measuring the side-channel leakage of ineffectively faulty ciphertexts. In particular, he measures the leakages $(l_i)_{0 \le i < N}$ of N ciphertexts $(c'_i)_{0 \le i < N}$ with respective HWs $(h_i)_{0 \le i < N}$, while faulting the 10th round SBox. The attack's goal is to use the side-channel and fault profiling steps described in Sections B.2.2.1 and B.2.2.2 in order to recover the secret key. In the following, we describe different key recovery strategies.

Distribution comparison. A straightforward key recovery strategy

consists of the following steps. First, the attacker simply classifies each leakage l_i into its mostly likely HW value, i.e. he finds the value h that maximizes the likelihood:

$$\Pr[h|l_i] \propto \Pr[l_i|h] \cdot \Pr[h] \tag{B.1}$$

where $\Pr[l_i|h] = \mathcal{N}(l_i|(\mu_h, \sigma^2))$ (estimated in B.2.2.1) and $\Pr[h] = 2^{-8} \cdot \binom{8}{h}$ is the prior probability of a HW h. Once all N ciphertext leakages have been classified, the attacker builds a histogram of the ciphertext's HW distribution that we denote by $\tilde{\mathcal{D}}_k$. Finally, a key candidate k^* is scored based on how similar its fault-based hypothetical distribution $\mathcal{D}_{k^*}$ (estimated in B.2.2.2) is to $\tilde{\mathcal{D}}_k$. Different distinguishers can be used to compare distributions, such as the squared euclidean imbalance [63, 50], χ^2 and Kullback-Leibler divergence [108].

Improved distribution comparison. An issue that arises when comparing the hypothetical distributions $\mathcal{D}_{k^*}$ to the empirical distribution $\tilde{\mathcal{D}}_k$, is that while the hypothetical distributions correspond to true values, $\tilde{\mathcal{D}}_k$ is estimated from side-channel leakages and is impacted by the side-channel noise. In the profiled case we investigate, it is possible to improve the previous comparison by accurately simulating the impact of the noise on the hypothetical distributions, altering slightly the process in B.2.2.2. First, we add to the ciphertext's Hamming weights (used to estimate the hypothetical distributions $\mathcal{D}_{k^*}$) noise samples, knowing the noise standard deviation σ, to simulate their side-channel leakage. Then each simulated leakage is classified into its most likely HW value based on the profiled leakage model of Section B.2.2.1. This makes distributions $\mathcal{D}_{k^*}$ more comparable to the empirical distribution $\tilde{\mathcal{D}}_k$ that is estimated from the side-channel leakage. Another approach to perform this, is to estimate a confusion matrix on the HW classification and multiply it by the hypothetical ineffective fault distributions.

Maximum likelihood. A more generic approach to this problem is to follow a maximum likelihood strategy that takes into account the full distribution of the leakage and not only the classified Hamming weights. In our case, we are interested in the likelihood of a key candidate k^*, having observed N leakages $(l_i)_{0 \leq i < N}$ of the biased ciphertexts. The

likelihood $\Pr[k^*|(l_i)_{0\leq i<N}]$ is evaluated as follows:

$$\Pr[k^*|(l_i)_{0\leq i<N}] = \prod_{i=0}^{N-1} \Pr[k^*|l_i] = \prod_{i=0}^{N-1} \frac{\Pr[l_i|k^*] \cdot \Pr[k^*]}{\Pr[l_i]} \tag{B.2}$$

$$\propto \prod_{i=0}^{N-1} \Pr[l_i|k^*] \propto \prod_{i=0}^{N-1} \sum_{h=0}^{8} \Pr[l_i|h] \cdot \Pr[h|k^*] \tag{B.3}$$

Equation B.2 assumes the independence of the ciphertexts' leakages and then applies Bayes' rule. Equation B.3 is simply deduced by ignoring the terms that do not depend on the key byte (i.e. do not help to distinguish the key candidates), in this case $\Pr[l_i]$, and terms constant for each key, in this case $\Pr[k^*] = \frac{1}{256}$, since the key byte is uniformly distributed without any prior knowledge on it. Then, Equation B.3 applies the law of total probabilities. In the final form, we recognize the term $\Pr[l_i|h]$ which simply corresponds to $\mathcal{N}(l_i|(\mu_h, \sigma^2))$ and was estimated in B.2.2.1. Similarly, we recognize the term $\Pr[h|k^*]$ which corresponds to the fault templates in B.2.2.2, i.e. $\Pr[h|k^*] = \mathcal{D}_{k^*}(h)$. This maximum likelihood derivation is reminiscent of the one used by Clavier and Reynaud [40] for their improved blind SCA using joint distributions of leakages.

B.3 Theoretical and simulated analysis

This section analyses the effectiveness of the proposed attack using simulations. In particular, Section B.3.1 discusses the necessary conditions on the fault distribution and investigates the impact of the side-channel leakage function and Section B.3.2 compares the previously described key recovery strategies.

B.3.1 Impact of the leakage function

The original conditions on SIFA also apply to the proposed attack. The first requirement for SIFA is a non-negligible ineffective fault probability. For instance, the uniform bit flip fault model does not fulfil such a condition [50]. The second SIFA requirement is that the injected fault results in a non-uniform distribution of ineffective faults. Accordingly, if we consider that the adversary has access only to the ciphertexts' Hamming weights, then a uniformly distributed fault implies that Hamming weights are distributed according to $\mathcal{B}(8, \frac{1}{2})$ (with $\mathcal{B}$ denoting the binomial distribution) for all key candidates, and thus the attacker is unable to distinguish the key byte.

In the following, we show that despite obtaining a fault model that conforms to the previous conditions, the success of blind side-channel SIFA does not only depend on the fault distribution but also on the deterministic part of the side-channel leakage. For the first part of this analysis, we leave aside the impact of the noise (which we investigate later) and mainly focus on the deterministic leakage function due to its inherent information compression: when a manipulated value is explicitly targeted by a side-channel adversary, only partial information can be recovered on it. This information can be represented by a surjective function (typically HW). A simple example that explicitly illustrates this point is a stuck-at-0 fault model, combined with a device leaking the HW of manipulated intermediates. This implies that the biased ciphertexts are equal to the key byte, since $c' = s' \oplus k = 0 \oplus k = k$. However, only the HW of the ciphertext (and thus of the key byte) can be observed and recovered through SCA. On average the number of remaining full keys to test after recovering the Hamming weights of its 16 bytes $\approx 2^{90}$.

While the stuck-at-0 case is simple to understand, it may be less trivial to notice the shortcomings of other induced biases in combination with the leakage function. For instance, despite inducing an appropriate bias for SIFA, a random-and fault combined with the HW function is not able to recover the full value of the key byte as demonstrated by the simulated experiments shown on Figure B.2. First, on the left side of Figure B.2, we plot the hypothetical distributions $\mathcal{D}_{k^*}$ of ciphertexts' Hamming weights biased by ineffective random-and faults, for all possible key byte values $k^* \in \mathbb{F}_{2^8}$. We can observe that some of these distributions are identical and thus cannot be distinguished. On the right side of Figure B.2 this observation is further confirmed. We performed the maximum likelihood attack described in Section B.2.2.3 on HW leakages (HW, blue line). We compare it to the same attack performed on identity leakages (ID, orange line), which corresponds to classical SIFA with full access to ineffective ciphertexts. The attacks are performed under negligible noise and we plot for both the average rank of the correct key byte as function of the number of fault inductions. Note that the ineffectivity rate (the ratio between the number of ineffective faults and the total number of fault injections) of a random-and fault is approximately 10%. Naturally, since random-and faults are suitable for SIFA, the attack succeeds using less than 50 ineffective faults. However, an identical attack performed on Hamming weights is unable to achieve an average rank below 20 candidates, regardless of the number of fault injections. This is due to the previously mentioned information compression property of the side-channel leakage function.

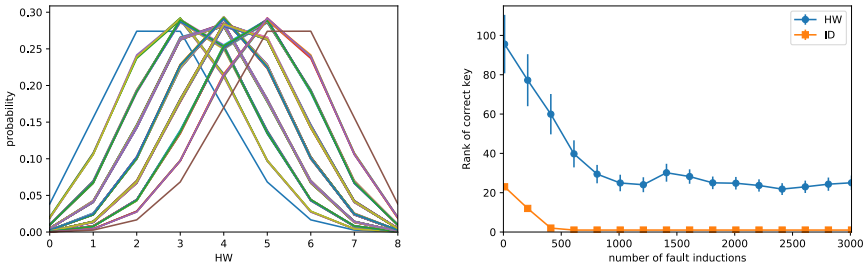


Figure B.2: Blind side-channel SIFA with random-and faults and noiseless HW leakage. Left: hypothetical distributions $\mathcal{D}_{k^*}$ for $k^* \in \mathbb{F}_{2^8}$. Right: comparison of classic SIFA and SIFA on Hamming weights for random-and faults.

The previous observations and simulated experiments suggest that to fully recover the key byte, the fault induction (together with the fault detection mechanism) is required to induce a suitable bias when combined with the leakage function. Specifically, the success of the attack depends on all possible hypothetical distributions (such as shown on the left side of Figure B.2) and the possibility to distinguish every distribution out of the 256 other ones. Notably, this condition is different from the one required for SIFA on the penultimate round which mainly depends on the advantage of distinguishing the biased distribution from a uniform one, and not distinguishing among many other distributions.

The suitability of an induced ineffective fault distribution can be easily confirmed by performing a regular SIFA (taking into account the leakage function) to evaluate the maximum information that can be recovered on the key byte, as shown for instance on the right side of Figure B.2. For instance, for random-and faults on average approximately 25 candidates cannot be distinguished after the attack.

B.3.2 Comparison of key recovery strategies

In this section, we compare the key recovery strategies described in Section B.2.2.3: the distribution comparison, the improved distribution comparison and the maximum likelihood strategy. In the following, to compare distributions we use the Kullback-Leibler divergence as a distinguisher which is defined from probability distribution $\mathcal{P}$ to $\mathcal{Q}$ as:

$$D_{KL}(\mathcal{P}||\mathcal{Q}) = \sum_x \mathcal{P}(x) \log \frac{\mathcal{P}(x)}{\mathcal{Q}(x)}$$

We also use a simulated ineffective fault distribution and simulated

HW leakages with noise standard deviation $\sigma \in \{0.7, 1.5\}$. The ineffective fault distribution is generated as a random highly non-uniform distribution on $\mathbb{F}_{2^8}$ as described by the following Python code:

```
fd = numpy.random.uniform(0.0,0.4,size=256)
fd = fd/numpy.sum(fd)
```

The distributions generated by this process usually allow to extract the full 8 bits of information on the key byte. We performed all three attacks for both noise levels and plot on Figure B.3 the average ranks they achieved as function of the number of ineffective faults. The left side corresponds to the low-noise case with $\sigma = 0.7$ leading to an SNR ≈ 4 and the right side to moderate-noise level with $\sigma = 1.5$ corresponding to an SNR ≈ 0.9. The x-axis corresponds to the number of ineffectively faulty ciphertexts and the y-axis to the rank of the key byte. The vertical lines on every data point correspond to the standard error on the mean estimation of the rank over the 200 experiments we performed.

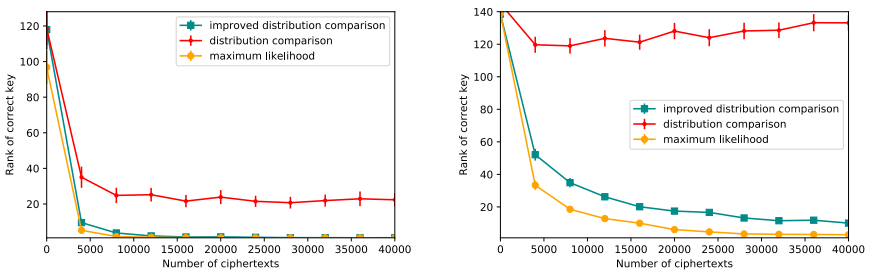


Figure B.3: Comparison of key recovery strategies for blind side-channel SIFA. Left: for low noise $\sigma = 0.7$, SNR ≈ 4. Right: for moderate noise $\sigma = 1.5$, SNR ≈ 0.9.

On Figure B.3, we observe that for low noise, the maximum likelihood approach and the improved distribution comparison achieve similar results, while the classical comparison is not able to decrease the ranks below 20 candidates. This is due to the fact that the empirical distribution $\tilde{\mathcal{D}}_k$ is distorted as a result of the side-channel noise, leading to a poor comparison with the hypothetical distributions $\mathcal{D}_{k^*}$. This observation is confirmed in the moderate noise case where the classical distribution comparison strategy is not able to distinguish the correct key, while both the improved comparison and the maximum likelihood strategies are able to sufficiently lower the key rank. However, we observe that the maximum likelihood attack is more robust against the noise and is the optimal key recovery strategy. Based on these results,

the following sections will focus on the maximum likelihood approach to carry out the attack.

B.4 Experimental validation

This section confirms the applicability of blind side-channel SIFA in an experimental setting, using a modern ARM micro-controller. We use a setup that injects faults while concurrently measuring the side-channel leakage. We use the same device for the profiling and the attack.

B.4.1 Target and setup description

The target device is an ARM Cortex-M4 based micro-controller running at a clock frequency of 32 MHz. The target implementation is the ANSSI AES-128 hardened library [18] protected with affine masking [64] against SCA. We disabled the shuffling countermeasure. While this implementation is not protected against faults, we simply assume that a redundant computation (duplication) is implemented in a masked manner as a fault countermeasure and incorrect ciphertexts are neither unmasked nor returned by the device. Instead, the unmasking of the ciphertext is followed by a state copy to return an array of bytes to the encryption function. To perform FI, a diode laser system is used to generate laser pulses with a diode current of 200 mA and a pulse width of 37 ns. The laser beam is focused on the die surface with a lens. During a χ^2-based preliminary parameters search, we found that a fault injection on the SRAM2 produces a significant bias on the output of the Sbox operation. For our experiments, we chose a fixed position within the SRAM2 block. A trigger and a delay generator were used to trigger the FI. Regarding side-channel measurements, a Langer microprobe was placed near the SRAM and the glue logic. The leakage traces of the target ciphertext byte were recorded with a Lecroy WR 625Zi oscilloscope at a sampling rate of 2.5 Ghz.

B.4.2 Hybrid attack: practical faults and simulated side-channel

Given the issues laid out in Section B.3, we first confirm that the ineffective fault distribution induced by the laser can indeed result in key recovery under a HW leakage function. Thus, we performed the maximum likelihood attack as described in Section B.2.2.3, using the ineffective fault distribution produced by the practical FI and we simulated the HW side-channel leakage. This hybrid attack is carried out for different

noise levels. It corresponds to ideal side-channel modeling, since we assume the leakage to conform strictly to the HW model. In addition, we use the same distribution to both estimate the hypothetical fault distributions and to sample the ineffective faults for the attack. The results are plotted on Figure B.4 where the x-axis corresponds to the number of ineffective faults and the y-axis to the average rank of the key bytes. The results shown on Figure B.4 demonstrate that the specific laser-induced faults, combined with noisy HW leakage of the ciphertexts can recover the correct value of the key byte in the context of SIFA-like attacks, without any input or output knowledge. It is also robust to the addition of side-channel noise and to the noise induced by failed (non-ineffective) fault injections during the experimental process.

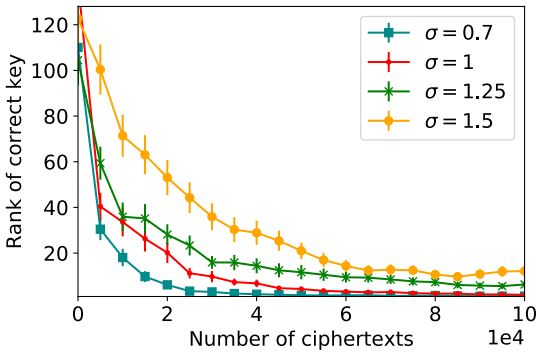


Figure B.4: Blind side-channel SIFA on ARM Cortex M4. The attack uses the estimated fault distributions after laser FI and simulated side-channel leakages.

B.4.3 Combined attack: practical faults and side-channel

Having established the stability of the attack we proceed with the results of the combined FI and SCA. Initially (profiling phase), the ineffective fault profiling was performed using random plaintexts and a random key as described in Section B.2.2.2. Continuing (attack phase), the combined FI and side-channel measurements were performed using random plaintexts and a fixed key. We note that our ineffective fault profiles did not exhibit any significant differences compared to the ones observed in the attack phase. We also emphasize that the issue of either fault or side-channel portability on different devices is orthogonal to this work. How this matter affects a combined attack such as performed in this work, however remains an interesting scope for further research.

Regarding the side-channel measurements, we obtained 150k traces corresponding to ineffective faults after 550k fault injections (leading to a 30% ineffectivity rate). We processed the traces by selecting (during profiling) the POI with the highest correlation to the HW of the ciphertext, then applied PCA [6] to compress the leakage further into a single dimension leading to an SNR ≈ 0.4. Due to the length of the experiment, we selected 40k traces from the middle of the experiment for profiling and used the remaining traces to conduct the maximum likelihood attack. Evaluation results are plotted on Figure B.5 as function of the number of ineffective faults.

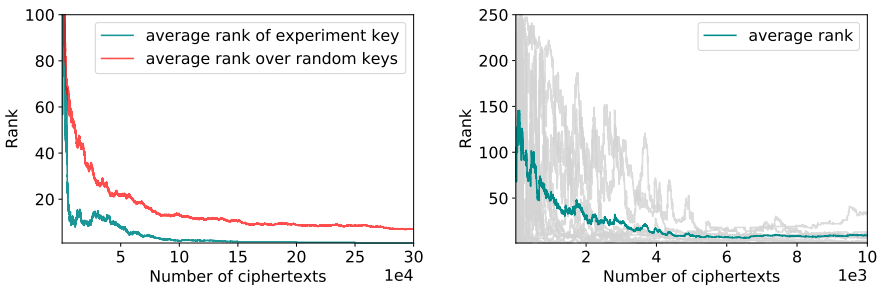


Figure B.5: Results of combined attack. (Right) Full combined experiment. (Left) Simulated side-channel with measurements' noise level.

First, on the right part of the figure we provide results of the full combined attack on a single key. The lowest average rank of 6.54 is achieved with less than 8k traces. To replicate the attack for a large number of keys, we use simulated side-channel attack with a noise level corresponding to the SNR observed on the device ($\sigma^2 = 5$). We show the results on the left part of Figure B.5, where we plot the average rank across random keys and additionally the average rank for the key used in the combined experiment. We notice that while the simulated attack eventually leads to an average rank of 1, the attack on real traces reaches rapidly a very low rank of 6.54 but is left with only a few candidates to distinguish. Presumably, this effect can be due to the experimental setup and the interaction between the side-channel leakage and the laser fault injection. This however was not verified experimentally. The thorough investigation of these effects and how to correct them is a very interesting scope for advanced research in the context of combined attacks.

B.5 Conclusion

This work considered profiled and combined FI and SCA in order to apply a SIFA-like key-recovery without any knowledge of the ciphertext. Concretely, we show that fault inductions on an ARM Cortex-M4 micro-controller lead to very low key ranks, despite only observing the side-channel leakage of ineffectively faulty ciphertexts, and without any plaintext repetition. However, our attack relies on both side-channel and fault profiling and additionally assumes that the ciphertext is an ephemeral secret that is unmasked after the fault detection mechanism.

Future work could potentially investigate the possibility of applying blind side-channel based SIFA in a fully unprofiled fashion, i.e. without any prior fault characterization or side-channel profiling, and also how this attack applies when infective countermeasures are deployed. Another direction for further research involves comparing the attack proposed in this work to blind high-order SCA, when the attacker has no control over or knowledge of the plaintext/ciphertext.

References

[1] Atsam4c-ek user guide : http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel_11251_SmartEnergy_ATSAM4C-EK-User_Guide_SAM4C8-SAM4C16_User-Guide.pdf.

[2] Cortex-m4 technical reference manual : http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439b/DDI0439B_cortex_m4_r0p0_trm.pdf.

[3] Reassure. http://reassure.eu/. Accessed: 2021-01-13.

[4] ISO/IEC JTC 1/SC 27. ISO/IEC 15408-1: Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model. International Organization for Standardization, Geneva, CH, 2009.

[5] Alexandre Adomnicai and Thomas Peyrin. Fixslicing aes-like ciphers new bitsliced AES speed records on arm-cortex M and RISC-V. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):402–425, 2021.

[6] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[7] Melissa Azouaoui, Davide Bellizia, Ileana Buhan, Nicolas Debande, Sébastien Duval, Christophe Giraud, Éliane Jaulmes, François Koeune, Elisabeth Oswald, François-Xavier Standaert, and Carolyn Whitnall. A systematic appraisal of side channel evaluation strategies. In Thyla van der Merwe, Chris J. Mitchell,

and Maryam Mehrnezhad, editors, *Security Standardisation Research - 6th International Conference, SSR 2020, London, UK, November 30 - December 1, 2020, Proceedings*, volume 12529 of *Lecture Notes in Computer Science*, pages 46–66. Springer, 2020.

[8] Melissa Azouaoui, François Durvaux, Romain Poussier, François-Xavier Standaert, Kostas Papagiannopoulos, and Vincent Verneuil. On the worst-case side-channel security of ECC point randomization in embedded devices. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *Progress in Cryptology - INDOCRYPT 2020 - 21st International Conference on Cryptology in India, Bangalore, India, December 13-16, 2020, Proceedings*, volume 12578 of *Lecture Notes in Computer Science*, pages 205–227. Springer, 2020.

[9] Melissa Azouaoui, Kostas Papagiannopoulos, and Dominik Zürner. Blind Side-Channel SIFA. Cryptology ePrint Archive, Report 2021/685, 2021. `https://eprint.iacr.org/2021/685`.

[10] Melissa Azouaoui, Romain Poussier, François-Xavier Standaert, and Vincent Verneuil. Key enumeration from the adversarial viewpoint. In Sonia Belaïd and Tim Güneysu, editors, *Smart Card Research and Advanced Applications - 18th International Conference, CARDIS 2019, Prague, Czech Republic, November 11-13, 2019, Revised Selected Papers*, volume 11833 of *Lecture Notes in Computer Science*, pages 252–267. Springer, 2019.

[11] Alessandro Barenghi, Luca Breveglieri, Israel Koren, Gerardo Pelosi, and Francesco Regazzoni. Countermeasures against fault attacks on software implemented AES: effectiveness and cost. In *Proceedings of the 5th Workshop on Embedded Systems Security, WESS 2010, Scottsdale, AZ, USA, October 24, 2010*, page 7. ACM, 2010.

[12] Lejla Batina, Lukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. Online template attacks. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2014.

[13] Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In Benedikt Gierlichs and

Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 23–39. Springer, 2016.

[14] Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal collision correlation attack on elliptic curves. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 553–570. Springer, 2013.

[15] Sonia Belaïd, Pierre-Évariste Dagand, Darius Mercadier, Matthieu Rivain, and Raphaël Wintersdorff. Tornado: Automatic generation of probing-secure masked bitsliced implementations. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 311–341. Springer, 2020.

[16] Sonia Belaïd, Dahmun Goudarzi, and Matthieu Rivain. Tight private circuits: Achieving probing security with the least refreshing. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 343–372. Springer, 2018.

[17] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020.

[18] Ryad Benadjila, Louiza Khati, Emmanuel Prouff, and Adrian Thillard. Hardened library for aes-128 encryption/decryption on arm cortex m4 architecture. https://github.com/ANSSI-FR/SecAESTM32.

[19] Daniel J. Bernstein, Tanja Lange, and Christine van Vredendaal. Tighter, faster, simpler side-channel security evaluations beyond computing power. *IACR Cryptol. ePrint Arch.*, 2015:221, 2015.

[20] Guido Bertoni, Luca Breveglieri, Israel Koren, Paolo Maistri, and Vincenzo Piuri. A parity code based fault detection for an implementation of the advanced encryption standard. In *17th IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2002), 6-8 November 2002, Vancouver, BC, Canada, Proceedings*, pages 51–59. IEEE Computer Society, 2002.

[21] Eli Biham. A fast new DES implementation in software. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 1997.

[22] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[23] Andrey Bogdanov, Ilya Kizhvatov, Kamran Manzoor, Elmar Tischhauser, and Marc Witteman. Fast and memory-efficient key recovery in side-channel attacks. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566 of *Lecture Notes in Computer Science*, pages 310–327. Springer, 2015.

[24] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptol.*, 14(2):101–119, 2001.

[25] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Em-*

bedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[26] Eric Brier and Marc Joye. Weierstraß elliptic curves and side-channel attacks. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 335–345. Springer, 2002.

[27] Olivier Bronchain, Gaëtan Cassiers, and François-Xavier Standaert. Give me 5 minutes: Attacking ASCAD with a single side-channel trace. *IACR Cryptol. ePrint Arch.*, 2021:817, 2021.

[28] Olivier Bronchain and François-Xavier Standaert. Side-channel countermeasures' dissection and the limits of closed source security evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):1–25, 2020.

[29] Olivier Bronchain and François-Xavier Standaert. Breaking masked implementations with many shares on 32-bit software platforms or when the security order does not matter. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):202–234, 2021.

[30] Bundesamt für Sicherheit in der Informationstechnik. Minimum Requirements for Evaluating Side-Channel Attack Resistance of Elliptic Curve Implementations. `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_ECCGuide_e_pdf`, 2016.

[31] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.

[32] Gaëtan Cassiers and François-Xavier Standaert. Towards globally optimized masking: From low randomness to low noise rate or

probe isolating multiplications with reduced randomness and security against horizontal attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):162–198, 2019.

[33] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[34] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[35] Marios O. Choudary and Pantelimon George Popescu. Back to massey: Impressively fast, scalable and tight security evaluation tools. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 367–386. Springer, 2017.

[36] Mathieu Ciet and Marc Joye. (virtually) free randomization techniques for elliptic curve cryptography. In Sihan Qing, Dieter Gollmann, and Jianying Zhou, editors, *Information and Communications Security, 5th International Conference, ICICS 2003, Huhehaote, China, October 10-13, 2003, Proceedings*, volume 2836 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2003.

[37] Christophe Clavier. Secret external encodings do not prevent transient fault analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 181–194. Springer, 2007.

[38] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware counter-

measures. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.

[39] 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 - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings*, volume 6476 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2010.

[40] Christophe Clavier and Léo Reynaud. Improved blind side-channel analysis by exploitation of joint distributions of leakages. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 24–44. Springer, 2017.

[41] Christophe Clavier, Léo Reynaud, and Antoine Wurcker. Quadrivariate improved blind side-channel analysis on boolean masked AES. In Junfeng Fan and Benedikt Gierlichs, editors, *Constructive Side-Channel Analysis and Secure Design - COSADE 2018*. Springer, 2018.

[42] EMV Co. Emv integrated circuit card specifications for payment systems, book 2, security and key management, november 2011. version 4.3.

[43] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.

[44] Jean-Sébastien Coron and Ilya Kizhvatov. An efficient method for random delay generation in embedded software. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume

5747 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2009.

[45] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In Jean-Jacques Quisquater and Bruce Schneier, editors, *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings*, volume 1820 of *Lecture Notes in Computer Science*, pages 277–284. Springer, 1998.

[46] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful mutual information and success rate in side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):49–79, 2019.

[47] Wouter de Groot, Kostas Papagiannopoulos, Antonio de la Piedra, Erik Schneider, and Lejla Batina. Bitsliced masking and ARM: friends or foes? In Andrey Bogdanov, editor, *Lightweight Cryptography for Security and Privacy - 5th International Workshop, LightSec 2016, Aksaray, Turkey, September 21-22, 2016, Revised Selected Papers*, volume 10098 of *Lecture Notes in Computer Science*, pages 91–109. Springer, 2016.

[48] A. Adam Ding, Liwei Zhang, Yunsi Fei, and Pei Luo. A statistical model for higher order DPA on masked devices. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 147–169. Springer, 2014.

[49] Christoph Dobraunig, Maria Eichlseder, Hannes Groß, Stefan Mangard, Florian Mendel, and Robert Primas. Statistical ineffective fault attacks on masked AES with fault countermeasures. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 315–342. Springer, 2018.

[50] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: exploiting ineffective fault inductions on symmetric cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):547–572, 2018.

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

[52] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.

[53] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.

[54] Margaux Dugardin, Louiza Papachristodoulou, Zakaria Najm, Lejla Batina, Jean-Luc Danger, and Sylvain Guilley. Dismantling real-world ECC with horizontal and vertical template attacks. In François-Xavier Standaert and Elisabeth Oswald, editors, *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, volume 9689 of *Lecture Notes in Computer Science*, pages 88–108. Springer, 2016.

[55] Vincent Dupaquis and Alexandre Venelli. Redundant modular reduction algorithms. In Emmanuel Prouff, editor, *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, volume 7079 of *Lecture Notes in Computer Science*, pages 102–114. Springer, 2011.

[56] François Durvaux and François-Xavier Standaert. From improved leakage detection to the detection of points of interests in leakage traces. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceed-*

ings, Part I, volume 9665 of *Lecture Notes in Computer Science*, pages 240–262. Springer, 2016.

[57] Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. Building a side channel based disassembler. *Trans. Comput. Sci.*, 10:78–99, 2010.

[58] Yunsi Fei, A. Adam Ding, Jian Lao, and Liwei Zhang. A statistics-based success rate model for DPA and CPA. *J. Cryptogr. Eng.*, 5(4):227–243, 2015.

[59] Yunsi Fei, Qiasi Luo, and A. Adam Ding. A statistical model for DPA with novel algorithmic confusion analysis. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 233–250. Springer, 2012.

[60] Wieland Fischer, Christophe Giraud, Erik Woodward Knudsen, and Jean-Pierre Seifert. Parallel scalar multiplication on general elliptic curves over f_p hedged against non-differential side-channel attacks. *IACR Cryptol. ePrint Arch.*, 2002:7, 2002.

[61] Pierre-Alain Fouque and Frédéric Valette. The doubling attack - *Why Upwards Is Better than Downwards*. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2003.

[62] Apostolos P. Fournaris, Louiza Papachristodoulou, Lejla Batina, and Nicolas Sklavos. Residue number system as a side channel and fault injection attack countermeasure in elliptic curve cryptography. In *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era, DTIS 2016, Istanbul, Turkey, April 12-14, 2016*, pages 1–4. IEEE, 2016.

[63] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. Fault attacks on AES with faulty ciphertexts only. In Wieland Fischer and Jörn-Marc Schmidt, editors, *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013*, pages 108–118. IEEE Computer Society, 2013.

[64] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *Selected Areas in Cryptography - SAC 2010*. Springer, 2010.

[65] Benoît Gérard and François-Xavier Standaert. Unified and optimized linear collision attacks and their application in a non-profiled setting. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 175–192. Springer, 2012.

[66] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.

[67] Richard Gilmore, Neil Hanley, and Máire O'Neill. Neural network based attack on a masked implementation of AES. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015*, pages 106–111. IEEE Computer Society, 2015.

[68] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2015.

[69] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side-channel resistance validation. In *Non-Invasive Attack Testing Workshop - NIAT 2011*, 2011.

[70] Daniel M. Gordon. A survey of fast exponentiation methods. *J. Algorithms*, 27(1):129–146, 1998.

[71] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and

Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[72] Dahmun Goudarzi, Anthony Journault, Matthieu Rivain, and François-Xavier Standaert. Secure multiplication for bitslice higher-order masking: Optimisation and comparison. In Junfeng Fan and Benedikt Gierlichs, editors, *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings*, volume 10815 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2018.

[73] 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 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 567–597, 2017.

[74] Dahmun Goudarzi, Matthieu Rivain, and Damien Vergnaud. Lattice attacks against elliptic-curve signatures with blinded scalar multiplication. In Roberto Avanzi and Howard M. Heys, editors, *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, volume 10532 of *Lecture Notes in Computer Science*, pages 120–139. Springer, 2016.

[75] Joey Green, Arnab Roy, and Elisabeth Oswald. A systematic study of the impact of graphical models on inference-based attacks on AES. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers*, volume 11389 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2018.

[76] Vincent Grosso and François-Xavier Standaert. Asca, SASCA and DPA with enumeration: Which one beats the other and when? In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security,*

Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II, volume 9453 of *Lecture Notes in Computer Science*, pages 291–312. Springer, 2015.

[77] Vincent Grosso and François-Xavier Standaert. Masking proofs are tight and how to exploit it in security evaluations. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 385–412, 2018.

[78] Qian Guo, Vincent Grosso, François-Xavier Standaert, and Olivier Bronchain. Modeling soft analytical side-channel attacks from a coding theory viewpoint. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):209–238, 2020.

[79] Neil Hanley, HeeSeok Kim, and Michael Tunstall. Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, volume 9048 of *Lecture Notes in Computer Science*, pages 431–448. Springer, 2015.

[80] Neil Hanley, Michael Tunstall, and William P. Marnane. Unknown plaintext template attacks. In Heung Youl Youm and Moti Yung, editors, *Information Security Applications, WISA 2009*. Springer, 2009.

[81] Helmut Hasse. Zur theorie der abstrakten elliptischen funktionenkörper iii. die struktur des meromorphismenrings. die riemannsche vermutung. *Journal für die reine und angewandte Mathematik*, 175:193–208, 1936.

[82] Daniel Heinz and Thomas Pöppelmann. Combined fault and DPA protection for lattice-based cryptography. *IACR Cryptol. ePrint Arch.*, 2021:101, 2021.

[83] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006.

[84] Naofumi Homma, Atsushi Miyamoto, Takafumi Aoki, Akashi Satoh, and Adi Shamir. Collision-based power analysis of modular exponentiation using chosen-message pairs. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2008.

[85] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *J. Cryptogr. Eng.*, 1(4):293–302, 2011.

[86] Michael Hutter and Peter Schwabe. Nacl on 8-bit AVR microcontrollers. In Amr M. Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors, *Progress in Cryptology - AFRICACRYPT 2013, 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24, 2013. Proceedings*, volume 7918 of *Lecture Notes in Computer Science*, pages 156–172. Springer, 2013.

[87] Michael Hutter and Erich Wenger. Fast multi-precision multiplication for public-key cryptography on embedded microprocessors. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 459–474. Springer, 2011.

[88] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[89] Tetsuya Izu and Tsuyoshi Takagi. A fast parallel elliptic curve multiplication resistant against side channel attacks. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 280–296. Springer, 2002.

[90] Kimmo Järvinen and Josep Balasch. Single-trace side-channel attacks on scalar multiplications with precomputations. In Ker-

stin Lemke-Rust and Michael Tunstall, editors, *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, volume 10146 of *Lecture Notes in Computer Science*, pages 137–155. Springer, 2016.

[91] Marc Joye and Christophe Tymen. Compact encoding of non-adjacent forms with applications to elliptic curve cryptography. In Kwangjo Kim, editor, *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, volume 1992 of *Lecture Notes in Computer Science*, pages 353–364. Springer, 2001.

[92] Marc Joye and Christophe Tymen. Protections against differential analysis for elliptic curve cryptography. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 377–390. Springer, 2001.

[93] Marc Joye and Sung-Ming Yen. The montgomery powering ladder. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2002.

[94] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on keccak. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):243–268, 2020.

[95] A. Karatsuba and Yu. Ofman. Multiplication of multidigit numbers on automata. *Soviet physics. Doklady*, 7:595–596, 1963.

[96] Tae Hyun Kim, Tsuyoshi Takagi, Dong-Guk Han, Ho Won Kim, and Jongin Lim. Side channel attacks and countermeasures on pairing based cryptosystems over binary fields. In David Pointcheval, Yi Mu, and Kefei Chen, editors, *Cryptology and Network Security, 5th International Conference, CANS 2006, Suzhou, China, December 8-10, 2006, Proceedings*, volume 4301 of *Lecture Notes in Computer Science*, pages 168–181. Springer, 2006.

[97] Donald E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms*. Addison-Wesley, 1969.

[98] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[99] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[100] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[101] Oliver Kömmerling and Markus G. Kuhn. Design principles for tamper-resistant smartcard processors. In Scott B. Guthery and Peter Honeyman, editors, *Proceedings of the 1st Workshop on Smartcard Technology, Smartcard 1999, Chicago, Illinois, USA, May 10-11, 1999*. USENIX Association, 1999.

[102] Boris Köpf and David A. Basin. An information-theoretic model for adaptive side-channel attacks. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 286–296. ACM, 2007.

[103] Roman Korkikian, Sylvain Pelissier, and David Naccache. Blind fault attack against SPN ciphers. In Assia Tria and Dooho Choi, editors, *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014*. IEEE Computer Society, 2014.

[104] Brian Koziel, A.-Bon Ackie, Rami El Khatib, Reza Azarderakhsh, and Mehran Mozaffari Kermani. Sike'd up: Fast and secure hardware architectures for supersingular isogeny key encapsulation. *IACR Cryptol. ePrint Arch.*, 2019:711, 2019.

[105] Tanja Lange, Christine van Vredendaal, and Marnix Wakker. Kangaroos in side-channel attacks. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th*

International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers, volume 8968 of *Lecture Notes in Computer Science*, pages 104–121. Springer, 2014.

[106] Duc-Phong Le, Chik How Tan, and Michael Tunstall. Randomizing the montgomery powering ladder. In Raja Naeem Akram and Sushil Jajodia, editors, *Information Security Theory and Practice - 9th IFIP WG 11.2 International Conference, WISTP 2015 Heraklion, Crete, Greece, August 24-25, 2015 Proceedings*, volume 9311 of *Lecture Notes in Computer Science*, pages 169–184. Springer, 2015.

[107] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. Power analysis attack: an approach based on machine learning. *Int. J. Appl. Cryptogr.*, 3(2):97–115, 2014.

[108] Yanis Linge, Cécile Dumas, and Sophie Lambert-Lacroix. Using the joint distributions of a cryptographic function in side channel analysis. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*, pages 199–213. Springer, 2014.

[109] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. How to estimate the success rate of higher-order side-channel attacks. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2014.

[110] David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.

[111] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2016.

[112] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[113] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Inf. Secur.*, 5(2):100–110, 2011.

[114] Daniel P. Martin, Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam. Counting keys in parallel after a side channel attack. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 313–337. Springer, 2015.

[115] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. Profiling power analysis attack based on MLP in DPA contest V4.2. In *39th International Conference on Telecommunications and Signal Processing, TSP 2016, Vienna, Austria, June 27-29, 2016*, pages 223–226. IEEE, 2016.

[116] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):348–375, 2020.

[117] David McCann, Carolyn Whitnall, and Elisabeth Oswald. ELMO: emulating leaks for the ARM cortex-m0 without access to a side channel lab. *IACR Cryptol. ePrint Arch.*, 2016:517, 2016.

[118] Marcel Medwed and Elisabeth Oswald. Template attacks on ECDSA. In Kyo-Il Chung, Kiwook Sohn, and Moti Yung, editors, *Information Security Applications, 9th International Workshop, WISA 2008, Jeju Island, Korea, September 23-25, 2008, Revised Selected Papers*, volume 5379 of *Lecture Notes in Computer Science*, pages 14–27. Springer, 2008.

[119] Nicolas Meloni. New point addition formulae for ECC applications. In Claude Carlet and Berk Sunar, editors, *Arithmetic of Finite Fields, First International Workshop, WAIFI 2007, Madrid, Spain, June 21-22, 2007, Proceedings*, volume 4547 of *Lecture Notes in Computer Science*, pages 189–201. Springer, 2007.

[120] Thomas S. Messerges. Securing the AES finalists against power analysis attacks. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000.

[121] Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

[122] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985.

[123] Nashwa A. F. Mohamed, Mohsin H. A. Hashim, and Michael Hutter. Improved fixed-base comb method for fast scalar multiplication. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *Progress in Cryptology - AFRICACRYPT 2012 - 5th International Conference on Cryptology in Africa, Ifrance, Morocco, July 10-12, 2012. Proceedings*, volume 7374 of *Lecture Notes in Computer Science*, pages 342–359. Springer, 2012.

[124] P. Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48:243–264, 1987.

[125] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[126] National Institute of Standards and Technology NIST. Lightweight Cryptography. `https://csrc.nist.gov/projects/lightweight-cryptography/`, 2021. [Online; accessed 1-June-2021].

[127] National Institute of Standards and Technology NIST. Post-Quantum Cryptography. `https://csrc.nist.gov/projects/post-quantum-cryptography/`, 2021. [Online; accessed 31-May-2021].

[128] Phong Q. Nguyen and Igor E. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Des. Codes Cryptogr.*, 30(2):201–217, 2003.

[129] National Institute of Standards and Technology. FIPS 140-3. Information Technology Laboratory, NIST, Gaithersburg, MD 20899-890.

[130] National Institute of Standards and Technology. FIPS 186-2: Digital signature standard (DSS). 2000.

[131] Louiza Papachristodoulou, Apostolos P. Fournaris, Kostas Papagiannopoulos, and Lejla Batina. Practical evaluation of protected residue number system scalar multiplication. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):259–282, 2019.

[132] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In David L. Waltz, editor, *Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA, USA, August 18-20, 1982*, pages 133–136. AAAI Press, 1982.

[133] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016.

[134] Romain Poussier, Yuanyuan Zhou, and François-Xavier Standaert. A systematic approach to the side-channel analysis of ECC implementations with worst-case horizontal attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 534–554, 2017.

[135] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 513–533. Springer, 2017.

[136] 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, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece,*

May 26-30, 2013. Proceedings, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[137] Mathieu Renauld and François-Xavier Standaert. Algebraic side-channel attacks. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Information Security and Cryptology - 5th International Conference, Inscrypt 2009, Beijing, China, December 12-15, 2009. Revised Selected Papers*, volume 6151 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2009.

[138] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2011.

[139] Matthieu Rivain. On the exact success rate of side channel analysis in the gaussian model. In *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, pages 165–183, 2008.

[140] Matthieu Rivain, Emmanuel Prouff, and Julien Doget. Higher-order masking and shuffling for software implementations of block ciphers. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.

[141] Thomas Roche, Victor Lomné, and Karim Khalfallah. Combined fault and side-channel attack on protected implementations of AES. In Emmanuel Prouff, editor, *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, volume 7079 of *Lecture Notes in Computer Science*, pages 65–83. Springer, 2011.

[142] Sayandeep Saha, Dirmanto Jap, Jakub Breier, Shivam Bhasin, Debdeep Mukhopadhyay, and Pallab Dasgupta. Breaking redundancy-based countermeasures with random faults and power

side channel. In *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2018, Amsterdam, The Netherlands, September 13, 2018*, pages 15–22. IEEE Computer Society, 2018.

[143] Sayandeep Saha, Debapriya Basu Roy, Arnab Bag, Sikhar Patranabis, and Debdeep Mukhopadhyay. Breach the gate: Exploiting observability for fault template attacks on block ciphers. *IACR Cryptology ePrint Archive*, 2019, 2019.

[144] Werner Schindler and Kouichi Itoh. Exponent blinding does not always lift (partial) spa resistance to higher-level security. In Javier López and Gene Tsudik, editors, *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, volume 6715 of *Lecture Notes in Computer Science*, pages 73–90, 2011.

[145] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.

[146] Werner Schindler and Andreas Wiemers. Power attacks in the presence of exponent blinding. *J. Cryptogr. Eng.*, 4(4):213–236, 2014.

[147] Werner Schindler and Andreas Wiemers. Generic power attacks on RSA with CRT and exponent blinding: new results. *J. Cryptogr. Eng.*, 7(4):255–272, 2017.

[148] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Mathematics of Computation*, 44:483–494, 1985.

[149] Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. A collision-attack on AES: combining side channel- and differential-attack. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 163–175. Springer, 2004.

[150] Claude E. Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28(4):656–715, 1949.

[151] Nigel P. Smart, Elisabeth Oswald, and Daniel Page. Randomised representations. *IET Inf. Secur.*, 2(2):19–27, 2008.

[152] Jerome A. Solinas. Generalized mersenne prime. In Henk C. A. van Tilborg, editor, *Encyclopedia of Cryptography and Security*. Springer, 2005.

[153] François-Xavier Standaert. How (not) to use welch's t-test in side-channel security evaluations. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers*, volume 11389 of *Lecture Notes in Computer Science*, pages 65–79. Springer, 2018.

[154] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[155] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2010.

[156] Adrian Thillard, Emmanuel Prouff, and Thomas Roche. Success through confidence: Evaluating the effectiveness of a side-channel attack. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2013.

[157] Michael Tunstall, Louiza Papachristodoulou, and Kostas Papagiannopoulos. Boolean exponent splitting. *IACR Cryptol. ePrint Arch.*, 2018:1226, 2018.

[158] Harshal Tupsamudre, Shikha Bisht, and Debdeep Mukhopadhyay. Destroying fault invariant with randomization - A countermeasure for AES against differential fault attacks. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 93–111. Springer, 2014.

[159] Jasper G. J. van Woudenberg, Marc F. Witteman, and Federico Menarini. Practical optical fault injection on secure microcontrollers. In Luca Breveglieri, Sylvain Guilley, Israel Koren, David Naccache, and Junko Takahashi, editors, *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2011, Tokyo, Japan, September 29, 2011*, pages 91–99. IEEE Computer Society, 2011.

[160] Vincent Verneuil. *Elliptic curve cryptography and security of embedded devices. (Cryptographie à base de courbes elliptiques et sécurité de composants embarqués)*. PhD thesis, University of Bordeaux, France, 2012.

[161] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2012.

[162] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2013.

[163] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873

of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

[164] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.

[165] Mathias Wagner. 700+ attacks published on smart cards: The need for a systematic counter strategy. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, volume 7275 of *Lecture Notes in Computer Science*, pages 33–38. Springer, 2012.

[166] Colin D. Walter. Sliding windows succumbs to big mac attack. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 286–299. Springer, 2001.

[167] B. L. Welch. The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[168] Carolyn Whitnall and Elisabeth Oswald. A cautionary note regarding the usage of leakage detection tests in security evaluation. *IACR Cryptol. ePrint Arch.*, 2019:703, 2019.

[169] Carolyn Whitnall and Elisabeth Oswald. A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In 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 256–284. Springer, 2019.

[170] Xin Ye, Thomas Eisenbarth, and William Martin. Bounded, yet sufficient? how to determine whether limited side channel informa-

tion enables key recovery. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 215–232. Springer, 2014.

[171] Çetin Kaya Koc. Analysis of sliding window techniques for exponentiation. *Computers and Mathematics with Applications*, 30:17–24, 1995.