# Side-Channel Analysis and Leakage-Resistance

François-Xavier Standaert

Crypto Group, ICTEAM Institute, UCLouvain, Belgium
e-mail: fstandae@uclouvain.be

Version 2.1, September 2026

License: Creative Commons BY-NC-ND 4.0.

# Preface

Modern "*cryptography is the scientific study of techniques for securing digital information, transactions and distributed computing*" [KL14]. Since the early nineteen-eighties, the good separation of duties between clearly stated and unambiguous assumptions on cryptographic primitives (that can only be falsified via cryptanalysis) and the security proof of complex schemes relying on these primitives has been the seed of numerous important advances. Most of these advances quantify the adversaries in terms of their computational power. In this setting, cryptographic proofs reduce the security of complex schemes to the security of simpler primitives, independently of the adversarial strategy and following Kerckhoff's main principle: "*The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience*".

The seminal work of Kocher et al. on Differential Power Analysis (DPA) came as a disruptive factor in this success story [KJJ99]. While it has been known for long that information security is broader than cryptography [Gol11], DPA shows that the first step when moving from mathematical abstraction towards applying cryptography (i.e., implementing cryptographic algorithms) is already prone to strong and hard to mitigate attacks. DPA has major consequences for cryptographic practice: protecting embedded devices against it is needed for any application where adversaries are granted with physical access to implementations.* DPA also has major consequences for cryptographic theory: capturing physical leakage requires re-thinking definitions, assumptions, designs and proofs [MR04]. Besides, while the treatment of modern cryptography mostly relies on complexity theory, understanding physical leakage requires broader tools, ranging from electronics and communications to information theory and statistical learning.

Kocher et al.'s results took the cryptographic research community and the embedded security industry by surprise. They gave rise to countless extensions and

* Remote physical attacks are increasingly documented too, see [MDB21] for a survey.

triggered fundamental changes in the way cryptographic implementations are analyzed. During the first decade following the DPA publication, researchers started to understand the most powerful attack vectors and to identify countermeasures susceptible to circumvent them. Yet, the conditions to make countermeasures effective were mostly informal and their evaluation was mostly based on engineering intuition. Mangard et al.'s book on power analysis attacks gives an excellent account of these heuristic times [MOP07]. Without better solutions to prevent the attacks, the industry reacted by combining countermeasures with a certain level of security by obscurity (i.e., keeping implementation details secret).

For a while, the security by obscurity paradigm made embedded security research and cryptographic research quite disconnected. One reason of this gap is that security by obscurity contradicts Kerckhoff's main principle and is inherently hard to state as a formal and precise assumption, which is instrumental in the mathematical treatment of modern cryptography.* However, research over the years showed that this situation is not inevitable. Progresses on the definition of cryptographic functionalities in the presence of leakage and on sufficient conditions for making countermeasures effective put forward a possibility to extend the treatment of modern cryptography to the physical setting [KR19]. As for the move from classical to modern cryptography, this evolution requires a change of viewpoint. Cryptographic definitions may look too strong at first glance, and (aim to) escape the specification of realistic adversarial strategies that has been driving the certification of cryptographic implementations so far [ABB+20]. Yet, leveraging the cryptographic methodology comes with the important benefit of a good separation of duties between physical assumptions to be confirmed thanks to empirical evidence (or falsified) and mathematical proofs demonstrating how these physical assumptions can be amplified to yield strong security guarantees. Concretely, it also allows taking advantage of continuous research advances on implementations, which a closed and time-limited evaluation process cannot.

The primary goal of this book is to combine theoretical and practical advances in embedded security into a consistent foundation for researchers and practitioners interested in secure cryptographic implementations. For this purpose, we primarily take the viewpoint of a designer/evaluator aiming to assess the worst-case security level of an implementation. We pay a special attention to identify the parts of our empirical evaluations that can be suboptimal and to discuss the tightness of our formal analyzes. We also justify this approach by showing that

* It turns out this paradigm is also the inverse of a success story, with several public reports of catastrophic failures with computational adversaries [Bog07b, GdKGM+08, DHW+12].

restricting adversaries beyond the formal assumptions required by a set of countermeasures is likely to lead to uncontrolled risks of security overstatements (i.e., a false sense of security). These choices are reflected by our choice of terminology: this book is about side-channel analysis, which is the general study of physical leakage in the context of cryptographic implementations, more than on side-channel attacks; it is also about leakage-resistance, which is the generic term we use to cover all the countermeasures that can (and must) be combined to secure these implementations. Interestingly, it turns out that the constructions that offer the best guarantees against worts-case adversaries can also be efficient.

In order to keep its content concise, the book follows three editorial choices.

First, side-channel attacks are a threat against all cryptographic implementations. The techniques we describe apply nearly identically to many symmetric-key algorithms, with similar levels of efficiency. As a result, we decided to illustrate them with a single block cipher. We selected the Advanced Encryption Standard (AES) for this purpose [DR20]. The AES is not the easiest algorithm to protect against side-channel attacks. As discussed in conclusions, the best solutions in the state-of-the-art are rather based on smaller S-boxes with optimized bitslice representations. Yet, it is by far the most investigated modern cipher, which makes it a natural candidate for illustration. Since the treatment of leakage-resistant modes of operations with tight security bounds strongly benefits from Tweakable Block Ciphers (TBCs) [LRW11], we also consider tweakable versions of the AES which rely on the same rounds and differ in their key scheduling parts. We refer to Appendix A for reminders about the AES cipher and its tweakable versions.

Second, the goal of this book is to establish the concepts needed to implement a cryptographic algorithm securely. As a result, it is not aimed to provide an exhaustive list covering all the best (still evolving) solutions of the state of the art. It rather aims to provide the theoretical and practical background which allows integrating these best solutions in various contexts and navigating through the side-channel analysis and leakage-resistance literature. For this purpose, we also provide the proof sketches of important results in order to put forward how formal models and tools can lead to convincing physical security arguments.

Finally, the way we aim to reach this goal is by connecting theoretical advances to cope with leakage and implementation techniques that can fulfill their physical assumptions. In order to preserve the global coherence of the text, we focus primarily on mode-level, primitive-level and implementation-level countermeasures which can be analyzed in sound mathematical models, and then discuss the practical relevance of these models. This leaves hardware-level counter-

measures out of our scope. As discussed in conclusions, this is not a statement about their importance. Hardware-level countermeasures are essential to keep leakage-resistant designs efficient. Yet, their analysis is specific to technological choices and we therefore decided to identify the general requirements they need to fulfill to be effective rather than to study their applications exhaustively.

Following these general guidelines, the book is structured in three parts.

# Contents

**Preface** **2**

**List of acronyms** **9**

**I Side-Channel Analysis** **11**

**Introduction** **12**

**1 Exemplary attack** **14**

**2 Evaluation tools** **20**
2.1 Measurement and pre-processing . . . . . 22
2.2 Detection and mapping . . . . . 24
2.3 Modeling and extraction . . . . . 34
2.4 Information exploitation . . . . . 44
2.5 Post-processing . . . . . 49

**3 Theoretical background** **55**
3.1 Side-channel analysis framework . . . . . 55
3.2 Bounds and shortcuts . . . . . 60
3.3 Computing the metrics . . . . . 63
3.4 Discussion and related works . . . . . 69

**Exercises** **75**

**II Leakage-Resistance 80**

**Introduction 81**

**4 Security definitions 83**
4.1 Completing the attacks' taxonomy . . . . . 83
4.2 Integrity with leakage . . . . . 88
4.3 Confidentiality with leakage . . . . . 89
4.4 Composite definitions . . . . . 92
4.5 Discussion and related works . . . . . 93

**5 Mode-level countermeasures 98**
5.1 Leakage-resilient MAC . . . . . 98
5.2 Leakage-resistant encryption . . . . . 103
5.3 Leakage-resistant AE . . . . . 108
5.4 Getting rid of DPA security . . . . . 111
5.5 Discussion and related works . . . . . 113

**6 Implementation-level countermeasures 118**
6.1 Shuffling (operations) . . . . . 118
6.2 Masking (data) . . . . . 123
6.3 Computing on masked data . . . . . 132
6.4 Combining countermeasures . . . . . 139

**7 Provably secure masking 140**
7.1 Threshold probing security . . . . . 140
7.2 Random probing security . . . . . 140
7.3 Noisy leakage security . . . . . 140
7.4 Discussion and related works . . . . . 140

**Exercises 141**

**III Secure Implementations 144**

**Introduction 145**

**8 Ensuring DPA security** **146**
8.1 DPA security in hardware . . . . . 146
8.2 DPA security in software . . . . . 146
8.3 Discussion and related works . . . . . 146

**9 Ensuring SPA security** **147**
9.1 SPA security in hardware . . . . . 147
9.2 SPA security in software . . . . . 147
9.3 Discussion and related works . . . . . 147

**10 Backwards security evaluations** **148**
10.1 Testing DPA security . . . . . 148
10.2 Testing SPA security . . . . . 148
10.3 Discussion and related works . . . . . 148

**Exercises** **149**

**Concluding remarks** **150**

**Bibliography** **150**

**Appendix A The Advanced Encryption Standard** **183**
A.1 Specifications . . . . . 183
A.2 Tweakable versions . . . . . 187

# List of acronyms

| | |
|---|---|
| AE | Authenticated Encryption |
| AES | Advanced Encryption Standard |
| ASIC | Application-Specific Integrated Circuit |
| ASSO | Absolute Statistical Security Order |
| BP | Belief Propagation |
| CIML | Ciphertext Integrity with Misuse and Leakage |
| CPA | Correlation Power Analysis |
| CCA | Chosen Ciphertext Attack |
| CCAmL | CCA with misuse-resilience and Leakage-resistance |
| DPA | Differential Power Analysis |
| DUT | Device Under Test |
| ENC | Private-key Encryption scheme |
| FPGA | Field Programmable Gate Array |
| HW | Hamming weight |
| IV | Initialization Vector |
| KDF | Key derivation function |
| KeS | Key Sensitive (variable) |
| KIS | Key and Input Sensitive (variable) |
| KR | Key Recovery |
| LRPM | Local Random Probing Model |
| LDA | Linear Discriminant Analysis |
| MAC | Message Authentication Code |
| MD | Message Distinguishing |
| MI | Mutual Information |
| PCA | Principal Component Analysis |
| PCB | Printed Circuit Board |
| PDF | Probability Density Function |
| PMF | Probability Mass Function |

| | |
|---|---|
| POI | Point-of-Interest (in a leakage trace) |
| PRF | Pseudo-Random Function |
| PRP | Pseudo-Random Permutation |
| PSO | Probing Security Order |
| RSI | Random Start Index |
| repSPA | Simple Power Analysis with repetition |
| RPM | Random Probing Model |
| SASCA | Soft Analytical Side-Channel Attack |
| SD | State Distinguishing |
| SNR | Signal-to-Noise Ratio |
| SPA | Simple Power Analysis |
| sUF-L | Strong Unforgeability with Leakage |
| TA | Template Attack |
| TAES1 | Tweakable AES with 128-bit tweak |
| TAES2 | Tweakable AES with 256-bit tweak |
| TGF | Tag Generation Function |
| TPM | Threshold Probing Model |
| TVLA | Test Vector Leakage Assessment |

# Part I

# Side-Channel Analysis

# Introduction

In this first part of the book, we aim to introduce side-channel analysis in a way that can support the introduction of sound countermeasures in the second part. As observed by Wagner already a couple of years ago, an exhaustive coverage of all the side-channel attacks published in the literature is no longer possible: we "*need a systematic counter strategy*" [Wag12]. This need is precisely what motivates the focus of this book on worst-case evaluations. For this purpose, we introduce a minimum set of tools that can be used to provide a good approximation (or bound) of the security level against the best possible attack vectors.

Concretely, we start with an exemplary side-channel analysis highlighting the different challenges that an evaluator faces in Chapter 1. We then use it to structure a 5-step evaluation workflow in Chapter 2, and introduce standard tools in order to perform each of the steps. We additionally show how to combine these tools in order to efficiently estimate the security level of an (unprotected) implementation. Eventually, some theoretical background behind side-channel analysis is given in Chapter 3. We first detail side-channel evaluation metrics, their estimation and bounds that can be used as shortcuts to speed up evaluations. We then discuss the optimality of our workflow's different steps and the need of profiling for worst-case evaluations. We conclude with a definition of side-channel attack that can serve as a basis for designing and analyzing countermeasures.

Overall, this first part of the book follows a bottom-up approach. We start from a concrete example to introduce the problem and list the different engineering challenges that it raises. We then gradually formalize side-channel analysis in order to pave the way towards a more (top-down) cryptographic treatment of leaking implementations, which will be the focus of the book's second part.

We additionally illustrate the concepts we introduce with experimental results. In order to simplify their understanding, we use so-called simulated measurements in the main chapters, where the leakage traces are generated according to a well-controlled model. This allows the examples to have intuitive expla-

nations that easily highlight the pros and cons of the different evaluation tools we describe. In order to also confront these tools to real-world conditions, we provide exercises at the end of the three sections, that mix paper-and-pencil and programming problems – the latter being based on actual side-channel measurements of an unprotected AES implementation in a 32-bit software device.

# Chapter 1

# Exemplary attack

The unavoidable weakness that is underlying all published side-channel attacks is that cryptographic computations are never instantaneous. So while classical security reductions assume adversaries who can intercept and possibly tamper with the inputs and outputs of cryptographic algorithms, Kocher et al.'s seminal work puts forward that physical measurements may leak information about their intermediate computations. It is easy to see that such a leakage can be devastating. Take for example the AES described in Appendix A, but limited to its first two operations (i.e., AddRoundKey and SubBytes). Rather than a large (128-bit) pseudorandom permutation, this reduced cipher can be viewed as the concatenation of sixteen small (8-bit) permutations. Therefore, the adversary can perform a key recovery attack by just targeting the key bytes one by one. The time complexity of this attack is $16 \times 2^8$ rather than $2^{16 \times 8}$ as it should be for the full-round AES.

In practice, side-channel leakage is usually not providing the exact outputs of a cipher's intermediate computations. Nevertheless, with the additional help of some statistical processing, even a modest leakage can be sufficient to break a cryptographic implementation using a similar divide-and-conquer approach.

Let us now set up the scene a bit more precisely by describing a realistic attack against an idealized 8-bit implementation of the AES. By 8-bit we mean that, at each clock cycle, the device in which the AES is implemented is able to perform an 8-bit operation. The attack essentially aims at exploiting information about the first cipher round that can be extracted by monitoring physical leakage like the power consumption [KJJ99] or the electromagnetic radiation [GMO01, QS01]. For example, a leakage trace $\boldsymbol{l}$ obtained by measuring the execution of the first two AES S-boxes is illustrated in Figure 1, for plaintext bytes $x_0$ and $x_1$, first round key bytes $k_0$ and $k_1$ and intermediate values $y_0, y_1$ (after the key additions) and

$z_0, z_1$ (after the S-boxes), where the subscripts correspond to the indexes of the manipulated bytes. This trace $\boldsymbol{l}$ contains $\tau = 300$ time samples, next denoted as $l_j$, and is expected to depend on the values of the $x_i$'s, $y_i$'s, $k_i$'s and $z_i$'s.

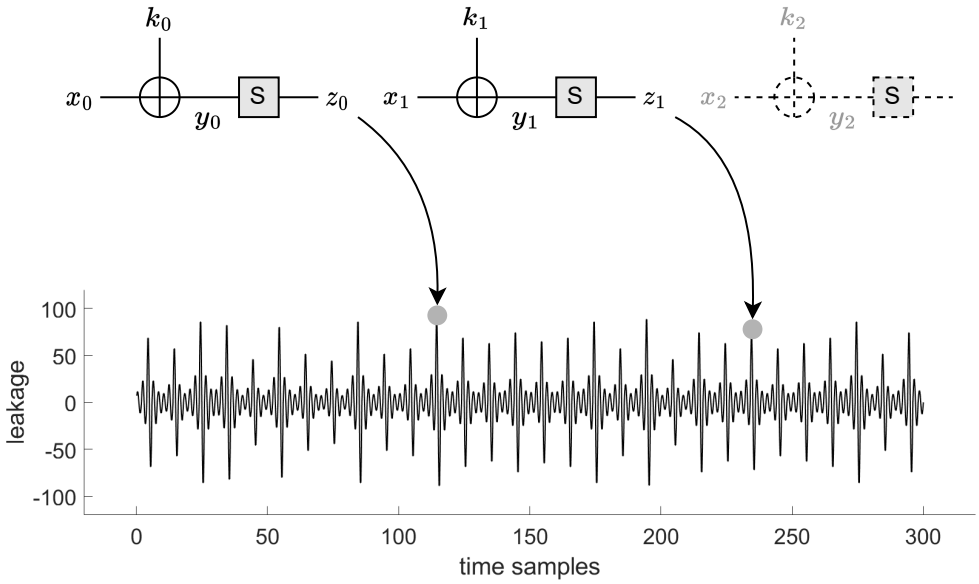



Figure 1: Leakage trace $\boldsymbol{l}$ for the AES' first two S-boxes ($\tau = 300$ samples).

In order to recover the secret key bytes, the side-channel adversary starts by defining a matrix of (e.g., random) plaintexts $\boldsymbol{x}$ of size $16 \times q$, corresponding to the 16 bytes of $q$ plaintexts. Taking advantage of the fact that the output (and therefore the leakage) of each S-box in the first AES round only depends on a single key byte, she can target every key byte independently. We next illustrate how to recover the first key byte $k_0$, which is XORed with the first plaintext byte $x_0$. An example of plaintext bytes vector $\boldsymbol{x}_0$ is given next for $q = 11$:

$$\boldsymbol{x_0} = [\,47,\ 231,\ 250,\ 112,\ 28,\ 66,\ 104,\ 152,\ 67,\ 154,\ 182\,].$$

Based on this vector of plaintext bytes, the adversary has to predict the leakage of the first S-box for different key candidates $k_0^* \in [0 : 255]$. Concretely, this implies computing ($q$-element) model vectors $\boldsymbol{m}^{z_0}(\boldsymbol{x_0}, k_0^*)$. Assuming that the

leakage of an intermediate variable is correlated with its Hamming weight, which is an acceptable approximation for unprotected CMOS devices [BCO04], it leads to $\boldsymbol{m}^{z_0}(\boldsymbol{x_0}, k_0^*) = \mathsf{HW}\big(\mathsf{S}(\boldsymbol{x_0} \oplus k_0^*)\big)$, with HW the Hamming weight function.

An example corresponding to the plaintext byte $\boldsymbol{x_0}$ above is given next:

$$\begin{aligned}
\boldsymbol{m}^{z_0}(\boldsymbol{x_0}, k_0^* = 0) &= [\,3,\ 3,\ 4,\ 3,\ 4,\ 3,\ 3,\ 3,\ 3,\ 4,\ 4\,], \\
\boldsymbol{m}^{z_0}(\boldsymbol{x_0}, k_0^* = 1) &= [\,3,\ 4,\ 4,\ 4,\ 3,\ 3,\ 6,\ 6,\ 3,\ 2,\ 4\,], \\
\boldsymbol{m}^{z_0}(\boldsymbol{x_0}, k_0^* = 2) &= [\,4,\ 5,\ 2,\ 1,\ 4,\ 2,\ 1,\ 4,\ 3,\ 3,\ 4\,], \\
&\ldots \\
\boldsymbol{m}^{z_0}(\boldsymbol{x_0}, k_0^* = 255) &= [\,3,\ 5,\ 5,\ 5,\ 2,\ 5,\ 2,\ 3,\ 4,\ 4,\ 5], \\
\end{aligned}$$

where the model values for the key candidate $k_0^* = 0$ equal $\mathsf{HW}\big(\mathsf{S}(47 \oplus 0)\big) = 5$, $\mathsf{HW}\big(\mathsf{S}(231 \oplus 0)\big) = 6$, $\mathsf{HW}\big(\mathsf{S}(250 \oplus 0)\big) = 6$, $\mathsf{HW}\big(\mathsf{S}(112 \oplus 0)\big) = 3$ ...

The next step is to measure the target device which contains the secret key byte to recover (e.g., $k_0 = 2$). For this purpose, the adversary collects leakage traces corresponding to the $q$ plaintexts of $\boldsymbol{x}$ and builds a leakage matrix $\bar{\boldsymbol{l}}(\boldsymbol{x}, k)$ of size $q \times \tau$. Let us temporarily assume the adversary knows that the S-box output value $z_0$ is manipulated at the time sample $115$. She can then reduce this matrix to a $q$-element leakage vector $\boldsymbol{l}_{115}(\boldsymbol{x}, k)$. Under the aforementioned (main) assumption that the Hamming weight model is accurate to a sufficient extent (which is necessary for the attack to succeed), this vector can be written as:

$$\boldsymbol{l}_{115}(\boldsymbol{x}, k) \approx [\,4,\ 5,\ 2,\ 1,\ 4,\ 2,\ 1,\ 4,\ 3,\ 3,\ 4\,] + \boldsymbol{r}_{115},$$

where $\boldsymbol{r}_{115}$ is an 11-dimension unknown random sampling, capturing the fact that actual measurements are generally affected by some physical noise.

At this stage, each model vector $\boldsymbol{m}^{z_0}(\boldsymbol{x_0}, k_0^*)$ can be viewed as $q$ samples of a univariate random variable $M^{z_0}(X_0, k_0^*) = \mathsf{HW}\big(\mathsf{S}(X_0 \oplus k_0^*)\big)$, where plaintext bytes are random and key byte candidates are fixed. Similarly, the leakage vector $\boldsymbol{l}_{115}(\boldsymbol{x}, k)$ can be viewed as $q$ samples of a univariate random variable $L(X, k)$. The adversary can therefore compare her 256 models with the actual measurements and select the key candidate that provides the best prediction. Pearson's correlation coefficient is a popular solution for this purpose [BCO04]:

$$\check{k}_0 = \underset{k_0^*}{\mathsf{argmax}}\ \hat{\rho}_q\big(L_{115}(X, k), M^{z_0}(X_0, k_0^*)\big)\,.$$

It can be estimated from $q$ leakage and model samples as follows:

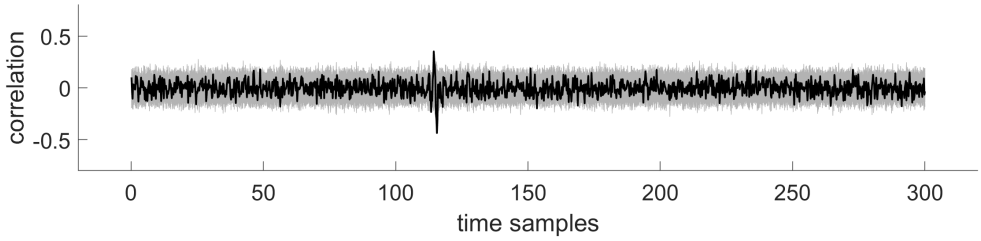
$$\hat{\rho}_q\left(L_j(X,k), M^{z_0}(X_0,k_0^*)\right) = \frac{\hat{\mathsf{E}}_q\left(L_j(X,k)\cdot M^{z_0}(X_0,k_0^*)\right) - \hat{\mathsf{E}}_q\left(L_j(X,k)\cdot \hat{\mathsf{E}}_q\left(M^{z_0}(X_0,k_0^*)\right)\right)}{\sqrt{\hat{\mathsf{Var}}_q(L_j(X,k))\cdot \hat{\mathsf{Var}}_q\left(M^{z_0}(X_0,k_0^*)\right)}},$$

with $\hat{\mathsf{E}}_q(.)$ and $\hat{\mathsf{Var}}_q(.)$ the sample mean and the sample variance, respectively.

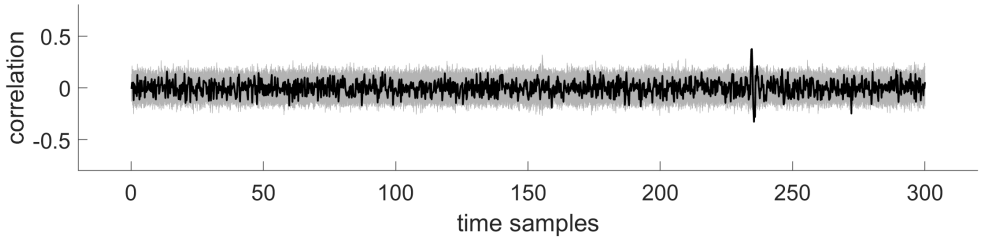
Finally getting rid of the temporary assumption that the adversary knows the S-box output value $z_0$ is manipulated at the time sample $115$, she can just estimate the correlation for each time sample independently and select the key as:

$$\check{k}_0 = \underset{j\in[0,\tau-1],\ k_0^*}{\mathsf{argmax}}\ \hat{\rho}\left(L_j(X,k), M^{z_0}(X_0,k_0^*)\right). \tag{1}$$

Figure 2 shows the result of such an attack for the key bytes $k_0$ and $k_1$, after the observation of $q = 256$ traces (rather than $q = 11$ in the example above).




(a) First S-box.



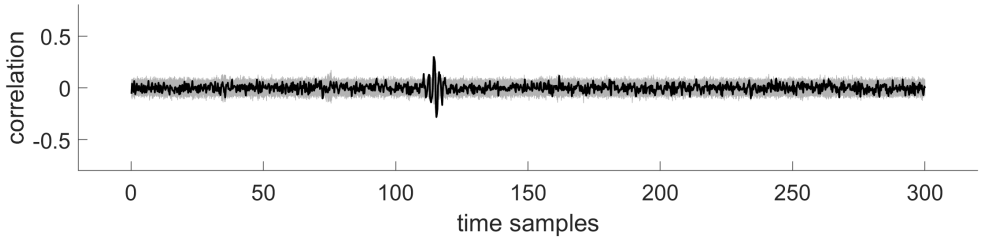

(b) Second S-box.

Figure 2: Correlation between the Hamming weights of the AES S-boxes' outputs ($\forall$ key byte candidates) and leakage traces for the correct key byte ($q = 256$).

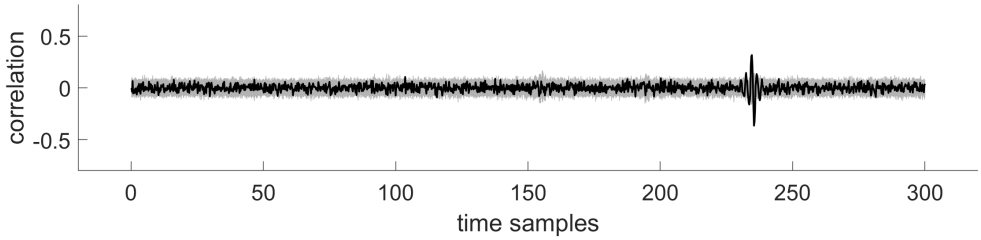
It can be observed that around time sample 115 (resp., 235), where the output of the first (resp., second) S-box is manipulated, the correlation estimated for the correct key byte (in black) is indeed higher than the one for the incorrect

key byte candidates (in grey). As illustrated in Figure 3, the gap between the correlation observed for the correct key byte and the incorrect key byte candidates can be increased by measuring more traces, since it allows a better estimation of the correlation coefficient. Yet, even in this case, the best estimated correlation values remain below 0.5, which can explained by the presence of additive noise in the traces or by slight model errors.* For uniformly distributed plaintexts, the same traces can be used to recover the other $k_i$'s (with $i \in [1:15]$) by adapting the models $\boldsymbol{m}^{z_i}(\boldsymbol{x_i}, k_i^*)$ and selecting key byte candidates as in Equation 1.

(*Exercise 1.1 is suggested in order to further illustrate the previous content.*)




(a) First S-box.




(b) Second S-box.

Figure 3: Correlation between the Hamming weights of the AES S-boxes' outputs ($\forall$ key byte candidates) and leakage traces for the correct key byte ($q = 1024$).

* Note that, as mentioned in introduction, the traces that we use in this part of the book are simulated in order to simplify their interpretation (actual side-channel measurements are provided in the exercises' parts). That is, for any target variable (e.g., $Z_i$) that we consider, we use a leakage function $\mathsf{L}_j(Z_i) \propto \delta(Z_i) + R_j$, with $\delta(.)$ a deterministic function of $Z_i$ and $R_j$ an additive Gaussian noise variable. Concretely, the function $\delta(.)$ was carefully chosen such that the correlation between its (noise-free) outputs and the outputs of the Hamming weight function equals 0.5, leaving room for more accurate models (i.e., models that better correlate with the leakage function) in the next chapters, where we always use the same leakage function.

Despite its conceptual simplicity, this attack already puts forward the main questions that side-channel adversaries and evaluators have to answer:

1. How to obtain good leakage traces (e.g., low-noise measurements)?
2. How to detect useful leakage samples, aka Points-of-Interest (POIs), and to map them to the intermediate computations they correspond to?
3. How to accurately model the useful leakage samples in order to extract (ideally all) the information contained in the measurements?
4. How to efficiently process the information extracted?
5. What if the information collected is not sufficient to recover the key?

In the following chapter, we will gradually build a side-channel evaluation toolbox by discussing and answering these five main questions one by one.

# Chapter 2

# Evaluation tools

The previous questions directly suggest the five-step side-channel analysis workflow depicted in Figure 4. We next use it as a way to structure the exposition of the various tools used in the evaluation of a leaking implementation. For this purpose, we detail state-of-the-art solutions to perform each of the steps, focusing on the tools that are most relevant for worst-case security evaluations.

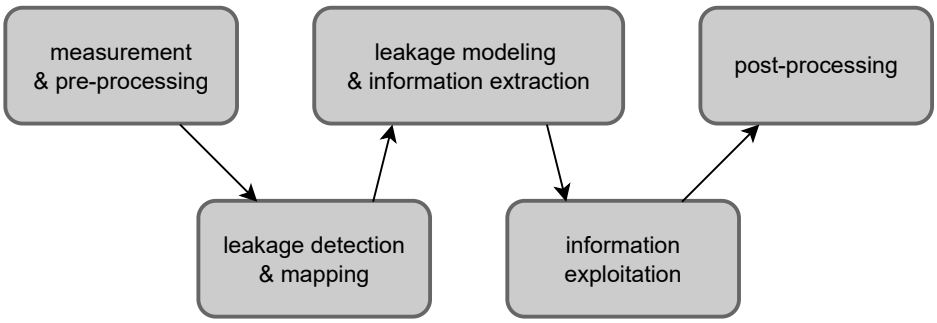



Figure 4: Side-channel analysis workflow.

As a preliminary, we need to define what we mean by worst-case evaluation since the use of this term in the side-channel research community deviates from its common acceptation in cryptography. In this respect, let us first recall that cryptographic research generally has two main goals. On the one hand, security proofs aim to reduce the security of complex functionalities (e.g., encryption, authentication in the case of symmetric cryptography) to the security of simpler primitives (e.g., block ciphers, permutations). On the other hand, cryptanalysis

aims to verify that the concrete schemes used to instantiate the primitives do not deviate from their expected behavior (i.e., match the proof assumptions). In this context, the term worst-case usually refers to the security guarantees provided by security reductions, which are independent of the adversarial strategy [KL14] and therefore hold even against an adversary leveraging an optimal strategy. By contrast, cryptanalysis is intrinsically hard to perform in a wort-case/optimal manner, as it corresponds to the art of combining different mathematical approaches in order to compromise the security of a scheme [Jou09, KR11]. Since side-channel analysis shares more similarities with cryptanalysis than with security reductions, one can therefore wonder what is the meaning of worst-case in this context?

The necessary clarification is that the term worst-case does not refer to the evaluation strategy (i.e., how the side-channel information is exploited) but to the means that evaluators are given to collect this information. A worst-case side-channel security evaluation, as defined next, corresponds to a context where all the information about the target implementation and all its intermediate computations are considered public during an offline learning (aka profiling) phase.

**Definition 1** (Worst-case side-channel security evaluation [ABB+20])**.** *A worst-case side-channel evaluation is one where the evaluator has unrestricted access to all the implementation details (e.g., software and hardware source codes) and to the secret keys and random values used by side-channel countermeasures (if they are implemented) during an offline learning (also called profiling) phase.*

The reason of this terminology relates to the fact that while the knowledge of the algorithms is now accepted as the standard model for mathematical cryptanalysis, the knowledge of the implementations remains a matter of debate in physical cryptanalysis. So while the term worst-case is synonym of security reductions offering guarantees against optimal adversaries in cryptography, it only refers to the fact that the evaluator has the best possible means to understand her target device in side-channel security evaluations. In other words, worst-case evaluations aim to help identifying optimal attack paths but they do not guarantee security against such (possibly unknown) optimal attack paths. We defer an assessment of the optimality of the steps in Figure 4 to Chapter 3, together with more justification of the practical relevance of this worst-case approach.

## 2.1 Measurement and pre-processing

Assembling a good measurement setup is the first step in the evaluation of a cryptographic implementation against side-channel analysis. A high-level view of its main components is given in Figure 5. Despite research papers usually contain a succint description of the tools they use for this purpose, there has been limited research specifically devoted to this topic. Chapter 3 of the DPA book remains a comprehensive source of information [MOP07]. It outlines a design space including the choice of a probing method (possibly embedding an amplifier), the supply voltage and clock frequency of the Device Under Test (DUT) and the resolution and sampling rate of the the evaluator's oscilloscope. Target implementations can be hardware: Field Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASICs), or software (e.g., embedded microcontrollers). Our focus in this book is on side-channel analyses that can be performed with relatively cheap equipment. Typically, commercial development boards for the DUTs with inductive probes and oscilloscopes in the Gsample/second range that can be gathered for a couple of thousands euros. We next discuss some basic recommendations to obtain the low-noise traces needed to perform efficient attacks [BUS21], and follow with a short discussion and a list of related works.

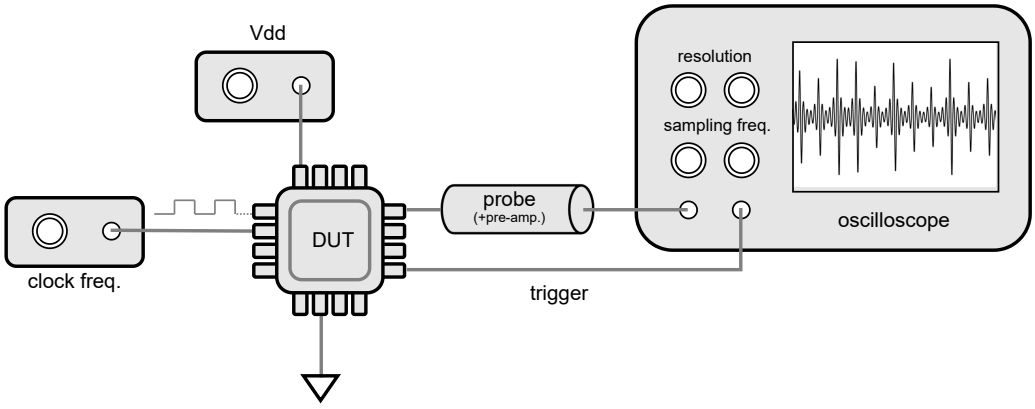



Figure 5: Side-channel measurement setup.

### 2.1.1 General guidelines

Regarding the DUT, we first mention that it is usually advisable to unsolder the decoupling capacitors from its Printed Circuit Board (PCB) in order to limit low-pass filtering effects. If possible, using an external crystal oscillator to derive the clock signal helps to limit the jitter-based noise. The clock frequency is then the most important parameter to set. A good rule-of-thumb is to use the highest available clock frequency that allows a visual identification of the cycles in the leakage traces. As for the supply voltage, it better comes from a stable (noise-free) source as well, but its value has a less significant impact and running the chip at nominal Vdd is in general sufficient to obtain good measurements.

Next, regarding the probe, standard options include the insertion of a small resistor in series with the power supply voltage and the use of an inductive probe. The latter is usually more convenient since it does not require any modification of the PCB. It is therefore the method of choice as long as the target leakage is covered by the probe's bandwidth. Low-noise pre-amplifiers with sufficient bandwidth can optionally be used in case of low-amplitude measurements.

As for the oscilloscopes, they have quite limited flexibility in terms of resolution (8-bit and 12-bit samples are standard values). The sampling rate is therefore the most critical parameter, since it directly affects the memory requirements of the security evaluations. As in general when quantizing signals, this sampling rate should at least be larger than the Nyquist frequency. Concretely, it is usually beneficial to increase this value slightly. A good rule-of-thumb (when the clock cycles are identifiable) is to take between 5 and 10 samples per clock cycle.

We note that in the worst-case security evaluation setting, we assume the availability of an accurate trigger for the target cryptographic computation.

### 2.1.2 Discussion and related works

Measurement setups are critical for side-channel security evaluations. They affect the noise level of the leakage samples, which is a key parameter for many countermeasures. Badly designed setups may artificially increase this noise level and lead to a false sense of security. The previous guidelines aim to mitigate this risk by enabling low-noise measurements without much additional pre-processing. They must be considered as an inherent part of a worst-case evaluation process, granting possibly unrealistic capabilities to the evaluator in order to gain security margins that can cover (hard to predict) engineering improvements. Note that the above guidelines are for standard setups exploiting the dynamic power

consumption. Setups targeting static power consumption require different optimizations (e.g., lower clock frequencies) in order to be exploitable [MMR20].

**Related works.** The descriptions above are for power measurements. Electromagnetic measurements lead to similar challenges with a different probe [AARR02]. Power and electromagnetic measurements can be combined into multi-channel attacks [ARR03]. When leaving the worst-case evaluation setting, concrete adversaries can face various additional challenges. A first one is the lack of accurate trigger, which may imply the need to rely to analog signals in order to synchronize the measurements [BBGV16]. Another one is the presence of noise in the frequency domain that may benefit from filtering (e.g., [PHF09, PS15]) and from analyses working directly in the frequency domain (e.g., [MRG$^+$10, TOT$^+$14]) or other transformed domains (e.g., [DSE$^+$12, UvWBS20]).

## 2.2 Detection and mapping

The results of Figures 2 and 3 show that a Correlation Power Analysis (CPA) allows an evaluator to identify the time samples where the two first AES S-box outputs are manipulated and their correct key byte at the same time. Despite convenient, this approach suffers from two limitations. First, it is non-profiled (hence not worst-case) and therefore it only works as long as the assumption that the leakage function is correlated with the Hamming weight model holds. Second, it is univariate (i.e., it exploits only one leakage sample at a time) and therefore may be suboptimal compared to approaches exploiting multiple samples jointly (e.g., in terms of number of leakage traces needed to recover the key). Both limitations suggest profiled and multivariate evaluations as a natural direction to better assess the worst-case security level of a leaking cryptographic implementation. Yet, they come with the additional burden of training a model during an offline phase, which can become expensive (e.g., in terms of number of leakage traces needed for profiling). Quantifying the cost and benefits of profiled analyses is therefore needed, which first requires defining the following complexities:

**Definition 2.** *(Profiling and attack complexities) We denote the number of side-channel traces collected (resp., the time needed) to estimate a leakage model in an offline phase as the profiling data (resp., time) complexity. We denote the number of side-channel traces (resp., the time) needed to perform a side-channel attack given a leakage model as the attack data (resp., time) complexity.*

As a result, profiling can be viewed as a way to trade an improved (online) attack complexity for some offline efforts. Since a straightforward multivariate es-

timation of the leakage can become expensive as the size of the traces increases, detection and mapping appeared as intermediate steps to make this tradeoff more appealing. Their goal is to identify the POIs in the leakage traces and to associate them with the intermediate computations that generate them. In machine learning terminology, they correspond to a feature selection step.

We next introduce three popular techniques that can be used for this purpose, discuss their specificities and respective advantages and detail how they can be used in the context of a worst-case side-channel security evaluation.

### 2.2.1 Signal-to-Noise Ratio

An intuitive solution to identify POIs in a leakage trace is the Signal-to-Noise Ratio (SNR) [Man04]. As its names suggests, it first requires to define what is the signal and what is the noise in the leakage of a target intermediate value. For this purpose, it is convenient to assume that leakage samples are the sum of a deterministic part and a noise part.* Say for example that we want to characterize the SNR of an S-box output $z_i$ in Figure 1, for which we posit:

$$l_j(z_i) = \delta_j(z_i) + r_j, \tag{2}$$

where $\delta_j$ is the leakage function's deterministic part at time sample $j$ and $r_j$ is an additive noise. Based on this we can directly define the SNR as:

$$\mathsf{SNR}(j) = \frac{\mathsf{Var}\big(\delta_j(Z_i)\big)}{\mathsf{Var}\big(R_j\big)}\,. \tag{3}$$

It remains to estimate $\delta_j(Z_i)$ and $R_j$. This can be be done by measuring several leakage traces for each value of $z_i$. Their sample mean can then serve as an estimation of the deterministic part $\hat{\delta}_j(z_i) = \hat{\mathsf{E}}\big(L_j(z_i)\big)$, and noise samples are just obtained by computing $\hat{r_j} = l_j(z_i) - \hat{\delta}_j(z_i)$, leading to an SNR estimate:

$$\hat{\mathsf{SNR}}_q(j) = \frac{\hat{\mathsf{Var}}_{256}\left(\hat{\mathsf{E}}_{\frac{q}{256}}\left(L_j(z_i)\right)\right)}{\hat{\mathsf{Var}}_q(\hat{R}_j)}\,, \tag{4}$$

where $\hat{\mathsf{E}}_\theta(.)$ and $\hat{\mathsf{Var}}_\theta(.)$ are the sample mean and variance with $\theta$ samples, respectively. In our AES case study, where the target intermediate value is 8-bit

* Which, as described in Chapter 1, is exactly satisfied by our simulated traces.

wide, the $q$ traces collected in order to estimate the signal variance are split in 256 sets of $\frac{q}{256}$ traces, corresponding to the 256 possible values of $z_i$. This definition of SNR assumes the additive noise to be independent of the data. If this assumption is incorrect, different noise variances can be computed for each value of $z_i$ and averaged, using 256 subsets of traces like for the signal estimation:

$$\hat{\mathsf{SNR}}_q(j) = \frac{\hat{\mathsf{Var}}_{256}\left(\hat{\mathsf{E}}_{\frac{q}{256}}\left(L_j(z_i)\right)\right)}{\hat{\mathsf{E}}_{256}\left(\hat{\mathsf{Var}}_{\frac{q}{256}}\left(L_j(z_i)\right)\right)}. \tag{5}$$

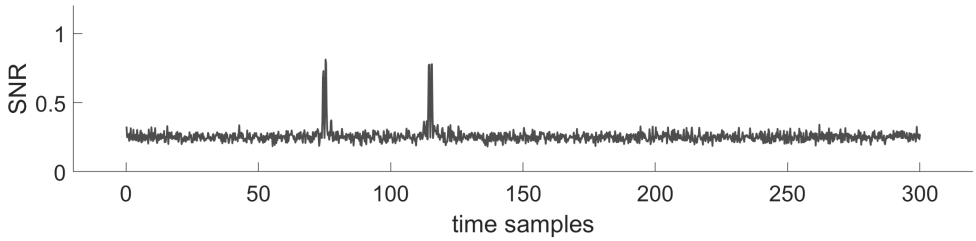
It is important to note that, contrary to Chapter 1, the leakage of the previous equations takes $z_i$ as argument rather than $x$ and $k$. This is because we specifically defined as signal the part of the leakage that depends on $z_i$. Concretely, the notation reflects that when computing a sample mean $\hat{\mathsf{E}}\big(L_j(z_i)\big)$, only $z_i$ should be fixed in the leaking device. For this purpose, the leakage traces must be measured with random plaintexts and keys and grouped according to the value of $z_i$ afterwards. Such a requirement emphasizes what may look as a somewhat arbitrary definition of signal. Indeed, nothing prevents adversaries to take advantage of the leakage that directly depends on the key as well. The corresponding SNR would be computed for samples $l_j(\$, k_i)$ where the plaintexts are random and a key byte is fixed. This would allow identifying the key scheduling intermediate computations, which motivates the following definition of sensitive variable:

**Definition 3** (Sensitive variables)**.** *We say that an intermediate value in a cryptographic implementation is Key and Input Sensitive (KIS) if it can be expressed as a function of a group operation (e.g., a XOR) between a part of the algorithm's secret key and a part of its public input (e.g., plaintext for a block cipher). We say that an intermediate value is Key Sensitive (KeS) if it can be expressed as a function of the algorithm's secret key independent of its public input.*

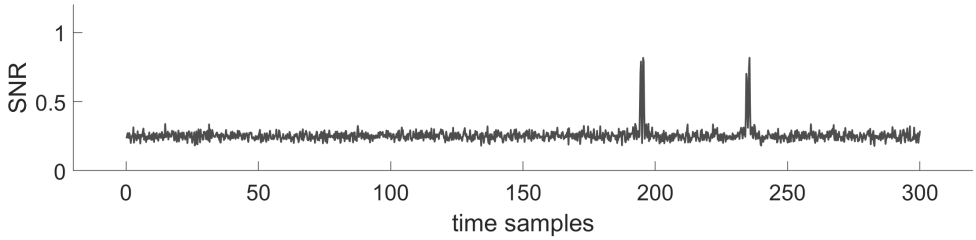
Clearly, the SNR of Equation 5 is for a KIS variable. We will see in Section 3.1.2 that such variables offer much stronger attack vectors than KeS variables.

An estimation of this SNR using $q = 1024$ traces, for the same case study as in Chapter 1, is given in Figure 6. It leads to two important observations.

First, both plots exhibit two peaks which correspond to the manipulation of the S-boxes inputs and outputs. This is expected since our definition of signal captures any leakage sample that is dependent on $\hat{\delta}_j(z_i)$. Since S-boxes are bijective, information on their inputs translates into information on their outputs.




(a) First S-box.




(b) Second S-box.

Figure 6: SNR for the AES S-boxes' outputs ($q = 1024$).

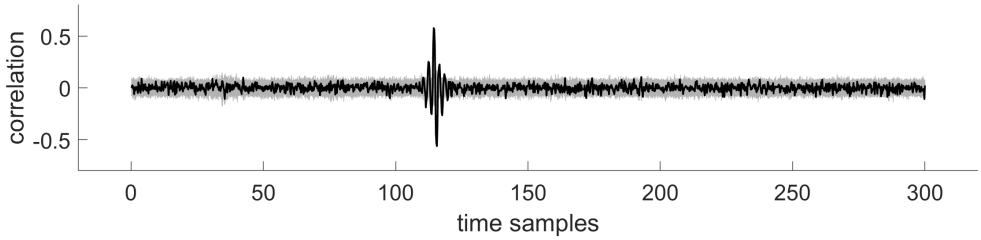
Second, the SNR is different from zero even for non-informative samples, as reflected by the "plateau" value of $\approx 0.25$ on the upper and lower plots. This relates to the fact that the SNR is an a priori metric. That is, it defines a priori that any difference in the mean values is signal. When combined with the estimation process of Equation 5, it implies that even estimation errors in the numerator mean values are considered as signal. Such an estimation therefore has a tendency to overestimate the SNR. Quite naturally, the bias due to signal estimation errors (which informally corresponds to the value of the plateau in Figure 6) is not statistically significant and vanishes as $q$ increases. The latter can be formally confirmed by computing a confidence interval for the SNR [Noe24].

(*Exercise 1.2 is suggested in order to further illustrate the previous content.*)

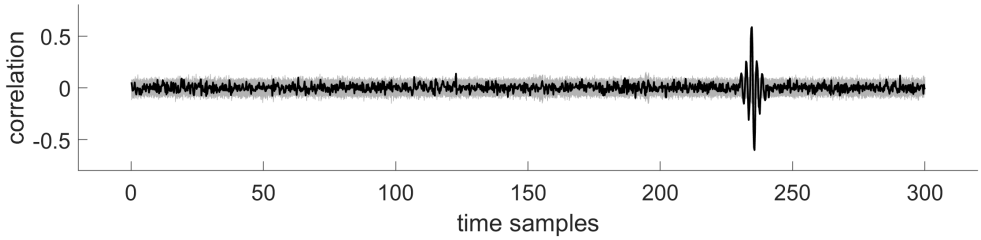
### 2.2.2 Profiled correlation

The previous definition of signal as a variance of leakage mean values suggests an extension of the CPA to the profiled setting that can serve as an alternative to the SNR. Remember that the attack of Chapter 1 uses the same Hamming weight model for all time samples: $m^{z_i}(x_i, k_i^*) = \mathsf{HW}\big(\mathsf{S}(x_i \oplus k_i^*)\big)$. A profiled CPA simply

replaces this Hamming weight assumption by an estimation of the mean leakage for the target intermediate value, evaluated for relevant (possibly all) time samples. Sticking with our S-box output example $z_i = \mathsf{S}(x_i \oplus k_i^*)$, it leads to model values $m_j^{z_i}(x_i, k_i^*) = \hat{\mathsf{E}}\big(L_j(z_i^*)\big)$. Note that this requires splitting the evaluators' set of $q$ traces into $q_\mathsf{p}$ profiling traces and $q_\mathsf{a}$ attack traces. The result of a profiled CPA using such models (perfectly) estimated for time samples $j = 115$ and $j = 235$ is given in Figure 7, leading to three important observation.




(a) First S-box.




(b) Second S-box.

Figure 7: Correlation between a profiled model for the AES S-boxes' outputs ($\forall$ key byte candidates) and leakage traces for the correct key byte ($q_\mathsf{a} = 1024$).

First, this detection is more specific than the SNR-based one as it only exhibits a single peak per S-box (vs. two in Figure 6). This is because the SNR metric differs from zero as soon as the 256 means corresponding to a leakage sample (for the 256 values of the target variable) differ. It is therefore preserved by the S-box (which only permutes these mean values). By contrast, in a CPA, the estimated correlation depends on the order of these means. Take for example the model vectors $\boldsymbol{m}^{z_i}(\boldsymbol{x_i}, k_i^*) = \mathsf{HW}\big(\mathsf{S}(\boldsymbol{x_i} \oplus k_i^*)\big)$ for a plaintext vector $\boldsymbol{x_i}$ with 256 elements ranging from 0 to 255. These model vectors contain the same 256 Hamming weights for all key byte candidates $k_i^*$, but in a different order, which is precisely what allows discriminating the correct key byte. So just as a wrong key

byte candidate has a permuted model vector that cancels the correlation with the actual measurements, the S-box permutes the model vector and prevents detecting samples at the input of the S-boxes when their output is predicted. Note that this specificity only holds if the same model is used for all time samples.

Second, the value of the correlation coefficient in Figure 7, which is estimated with the same total number of traces as the SNR in Figure 6, is stuck at $\approx 0$ for the uncorrelated samples. This is because, contrary to the SNR metric (which is a priori), the CPA embeds a cross-validating step. That is, when performing the profiled CPA, the models (which correspond to leakage mean values estimated offline with $q_p$ profiling traces) are tested against $q_a$ fresh attack traces, so that the estimation errors of these mean values can only lead to a reduction of the correlation coefficient. Yet, this difference vanishes as $q, q_{\mathsf{p}}$ and $q_{\mathsf{a}}$ increase and it can be shown that the SNR and the correlation coefficient computed for a profiled model carry the same asymptotic intuition (when $q_{\mathsf{p}}, q_{\mathsf{a}} \rightarrow \infty$ with $q = q_p + q_a$). More precisely, denoting a profiled model for time sample $j$ as $M_j$ and ignoring the random variable's arguments for simplicity, we have [Man04]:

$$\rho(M_j, L_j) = \frac{1}{\sqrt{1 + \frac{1}{\mathsf{SNR}(j)}}}. \tag{6}$$

Note that this result only holds for the correlation profiled per sample. Otherwise, we have no guarantees that the CPA exploits all the side-channel signal.

Finally, the value of the maximum correlation coefficient in Figure 7 is significantly increased compared to the unprofiled CPA of Figure 3. This confirms that the Hamming weight assumption, despite sufficient to perform a successful key recovery, was not perfect and did not allow an optimal univariate attack.

Interestingly, this improvement of the leakage model thanks to profiling can be characterized with the following correlation chain rule [SPRQ06]:

**Lemma 1** (Correlation chain rule)**.** *Let $X$, $Y$ and $Z$ be three random variables such that $Y = X + R_1$ and $Z = Y + R_2$ with $R_1$ and $R_2$ two additive Gaussian noise variables. Then, we have $\rho(X, Z) = \rho(X, Y) \cdot \rho(Y, Z)$.*

As a result, if we assume that an unprofiled model $M_j^{\mathsf{u}}$ corresponds to a profiled model $M_j^{\mathsf{p}}$ with additive model errors $M_j^{\mathsf{e}}$ and that we write the noisy leakage variable as a sum $L_j = (M_j^{\mathsf{u}} + M_j^{\mathsf{c}}) + R_j$, with $M_j^{\mathsf{c}} = -M_j^{\mathsf{e}}$ the model correction term and $R_j$ the noise, both assumed to follow a Gaussian distribution, so that $M_j^{\mathsf{u}} + M_j^{\mathsf{c}} = M_j^{\mathsf{p}}$, then $\rho(M_j^{\mathsf{u}}, L_j) = \rho(M_j^{\mathsf{u}}, M_j^{\mathsf{p}}) \cdot \rho(M_j^{\mathsf{p}}, L_j)$. In other words,

computing the correlation between a non-profiled leakage model and its profiled counterpart allows anticipating by which factor the correlation coefficients estimated in the non-profiled and profiled settings will differ. For example, the correlation between the profiled model of Figure 7 and the Hamming weight model of Figure 3 is $\approx 0.5$, which is confirmed by the peak values of these figures.

### 2.2.3 Welch's T-test

The SNR metric and profiled CPA are specific detection tools in the sense that they identify POIs that depend on specific intermediate computations. For this purpose, they estimate sample means for all the values of these target computations. As discussed in the previous section, the profiled CPA can be even more specific than the SNR as (in case the same profiled model is used for all the time samples of a trace) it allows distinguishing bijectively connected intermediate values. Quite naturally, it is also possible to go the other way and to design less specific detection tools. The standard approach for this purpose is to use Welch's T-test, and was introduced under the name Test Vector Leakage Assessment methodology (TVLA) [GJJR11]. Its original motivation was to provide a cheaper detection method which can possibly serve as a basis for conformance testing.

A natural way to make detection more efficient (in terms of data complexity) is to reduce the number of sample means to estimate. This is exactly what non-specific T-tests enable. Let us for example define the following estimates:

$$\begin{array}{rcl} \hat{\mu}_{\mathsf{f}}(j) = \hat{\mathsf{E}}_{q_{\mathsf{f}}}\big(L_j(x_{\mathsf{f}}, k_{\mathsf{f}})\big), & \hat{\sigma}^2_{\mathsf{f}}(j) & = \hat{\mathsf{Var}}_{q_{\mathsf{f}}}\big(L_j(x_{\mathsf{f}}, k_{\mathsf{f}})\big), \\ \hat{\mu}_{\mathsf{r}}(j) = \hat{\mathsf{E}}_{q_{\mathsf{r}}}\big(L_j(\$, \$)\big), & \hat{\sigma}^2_{\mathsf{r}}(j) & = \hat{\mathsf{Var}}_{q_{\mathsf{r}}}\big(L_j(\$, \$)\big), \end{array}$$

where $(x_{\mathsf{f}}, k_{\mathsf{f}})$ is a fixed pair of plaintext and key, $\$$ denotes uniformly random values and $q_{\mathsf{f}}$ and $q_{\mathsf{r}}$ respectively denote the number of samples used to estimate the mean and variance of the fixed and random leakage classes (for which a typical choice is $q_{\mathsf{f}} = q_{\mathsf{r}} = \frac{q}{2}$). We can then define the following T statistic:

$$\hat{\Delta}_{\frac{q}{2}}(j) = \frac{\hat{\mu}_{\mathsf{f}}(j) - \hat{\mu}_{\mathsf{r}}(j)}{\sqrt{\frac{\hat{\sigma}^2_{\mathsf{f}}(j)}{q_{\mathsf{f}}}} + \sqrt{\frac{\hat{\sigma}^2_{\mathsf{r}}(j)}{q_{\mathsf{r}}}}}. \tag{7}$$

The estimation of such a T-statistic for all the time samples of the same leakage traces as in our previous examples is illustrated in Figure 8. We use the popular rule-of-thumb which assumes POIs to be detected if the T-statistic is higher than a

threshold value of 4.5. It corresponds to a probability to reject the null hypothesis that there is no difference of mean between the fixed and random leakage sets below $10^{-5}$.* This figure leads to the following important observations.

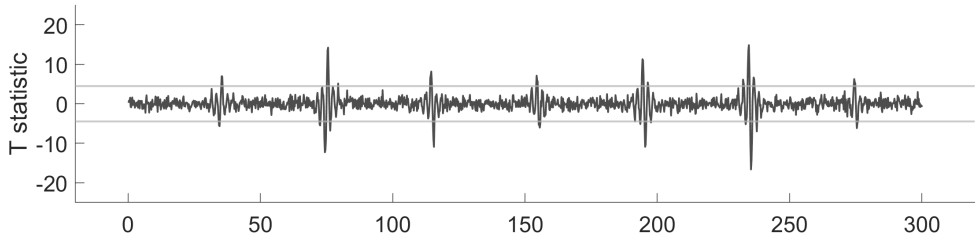



Figure 8: Non-specific T-test for the traces of Figure 1 ($q = 1024$).

First, and as expected, the detections are not specific: any leakage sample that depends on $x$ or $k$ can lead to a peak of the T-statistic. This implies that the POIs identified may not be easily exploitable in later evaluation steps (e.g., with a divide-and-conquer approach), since the value that generates their leakage may be hard to guess (e.g., if they correspond to the inner AES rounds). It also implies that such an approach does not help much to map the detected time samples with the operations of the implementation that generate their leakage.

Second, the value of the peaks differs for different operations while they were similar in Figures 6 and 7. This can be explained since the pair $(x_\mathrm{f}, k_\mathrm{f})$ may lead to intermediate computations of which the leakage mean value may be more or less close to the leakage mean value of the random class. For example, in the Hamming weight case with 8-bit intermediate computations, the random class has a mean leakage of 4. Therefore, any intermediate computation generated by the pair $(x_\mathrm{f}, k_\mathrm{f})$ that has a Hamming weight of 4 will be undetectable with Equation 7, whereas a Hamming weight value of 0 or 8 will be the easiest to detect. In other words, the non-specific T-test is incomplete when launched as a stand-alone tool: there are POIs that may not be identified. The TVLA methodology therefore advocates for running such a T-test for multiple pairs of plaintext and key. The more pairs are used, the more complete the test is, but the more its data complexity becomes similar to a specific detection using the SNR or a profiled CPA.

(*Exercise 1.3 is suggested in order to further illustrate the previous content.*)

* Formally, this probability is only correct for a univariate detection test and the threshold should therefore be adapted in function of the length of the leakage traces [DZD+17].

### 2.2.4 Discussion and related works

*Which tool to use?* The previous descriptions put forward that detection tools can be viewed as a tradeoff between different features: specific vs. non-specific (i.e., able to distinguish the leakage of different operations with more or less granularity), complete (i.e., able to capture the signal of all intermediate computations) or not, cross-validated or a priori.* As illustrated in Table 1, specific, complete and cross-validated tests tend to be more expensive (in data and in time).

| detection tool | *specific* | *complete* | *cross-validated* | *complexity* |
|---|---|---|---|---|
| SNR | v | v | x |  |
| profiled CPA | V | v | v |  |
| fixed vs. random T-test | x | x | x |  |

Table 1: Informal classification of detection tools.

In the context of a worst-case side-channel security evaluation, the detection and mapping step usually remains relatively cheap (since it mostly requires to estimate mean values). It can therefore be performed even for long traces and the reduced complexity of non-specific tests is rarely needed. Following the goal to improve the profiling complexity of an implementation, it also appears that the SNR metric provides just the level of specificity needed to detect the POIs that will be useful when modeling an intermediate value. More precisely, profiled leakage models typically try to take advantage of all the leakage samples that depend on a target intermediate computation. Hence, the possibility to detect dependent (e.g., bijectively connected) computations is an asset for this purpose.

*How to use detection?* The previous tools can be used for two distinct goals. Either for detection only, as part of a conformance-style test, or for detection and mapping in the context of an evaluation-style (possibly worst-case) analysis. In the first context, passing the test requires that no leakage is detected according to some threshold. It is then important to set the threshold in a statistically

* An interesting feature of complete tools like the SNR and the profiled CPA is that their value can be used as an indicator of the corresponding attack complexity – see Section 3.2.1.

sound manner. This requires computing the $p$-value of the null hypothesis that assumes no detection which, as mentioned in Section 2.2.3, can be challenging when traces become highly multivariate. Besides, even when carefully implemented, such conformance-style tests are prone to errors and have limited connection with the actual security level of leaking implementation [Sta18, WO19]. Acknowledging that "*absence of evidence is not evidence of absence*", a worst-case security analysis takes a different approach and does not consider a lack of detection as conclusive. Quite the opposite, it builds on detectable leakage, tries to to extract all its information and to exploit it optimally, so that sound countermeasures can in turn either reduce this information or make it hard to exploit. Detection is then combined with mapping and both serve as an intermediate step in the workflow of Figure 4. In this context, the data complexity will be increased until POIs can be identified thanks to visual inspection, making the discussion of statistical thresholds less critical since more quantitative steps follow.

We finally note that the descriptions in this (and the following) section(s) takes the selection of the target intermediate variables to exploit in a side-channel analysis for granted. In the worst-case setting we focus on in this book, identifying such (potentially) exploitable variables can indeed be done by inspecting the source codes and netlists that are assumed to be available. Besides, the leakage of an implementation (e.g., in hardware) may depend on the transitions between the consecutive values taken by a target intermediate variable (e.g., in a register) rather than on these values considered independently [SÖP04]. Identifying the (potentially) exploitable transitions in an implementation can also be done by inspecting source codes and netlists. Evaluators can therefore evaluate the signal that can be extracted using both value-based and transition-based models.

**Related works.** Leakage detection has been a topic of intense research, especially in the context of conformance testing. Despite not directly needed for our following worst-case investigations, we next point out a number of related works that can serve as complementary readings. Several authors investigated detection with different statistical tools. For example, [MOBW13] uses the mutual information, [BDGN14] the normalized inter-class variance and [BSS19] considers multivariate extensions of the T-test and the assumptions it requires. Other papers focus on optimizing detection. For example, [RGV17] improves its time complexity and [YTM$^+$20] improves its data complexity thanks to bootstrapping. The connection between conformance-style testing and evaluation-style testing is discussed in [RBG$^+$19]. The impact of the leakage partitioning in non-specific T-tests is examined in [DS16], which shows that fixed vs. fixed tests can lead to lower data complexitites than fixed vs. random ones, and [UvWBS20], which describes a one-hot partitioning

that is suitable to analyze parallel hardware implementations. Eventually, a large body of work focuses on higher-order analyses (e.g., [RGV12, SM16, MRSS18]), possibly taking advantage of deep learning [MWM21]. We defer their discussion to Part III of the book, where we discuss leakage detection in the context of protected implementations.

## 2.3 Modeling and extraction

The previous section described standard tools in order to identify POIs in leakage traces. We now describe how to profile these leakage samples and to use statistical models in order to extract information from them. As previously, we focus on tools that are useful in the context of a worst-case security evaluation.

### 2.3.1 Gaussian templates

The first profiled side-channel attack published in the literature was coined Template Attack (TA) and was introduced by Chari et al. [CRR02]. It remains among the most popular and useful tools for evaluators. In its basic form, a TA models the leakage samples with a Gaussian distribution. As any profiled evaluation tool, it works in two steps (modeling and extraction) that we detail next:

**Modeling.** Suppose the evaluator is provided with $q_{\mathsf{p}}$ profiling traces. The Gaussian TA models them with the Probability Density Function (PDF):

$$\mathsf{f}(\boldsymbol{l}|z_i) = \frac{1}{(2\pi)^{\frac{\tau}{2}}|\boldsymbol{\Sigma}_{z_i}|} \cdot \exp\left(-\frac{1}{2}(\boldsymbol{l}-\boldsymbol{\mu}_{z_i})^{\mathsf{T}} \cdot \boldsymbol{\Sigma}_{z_i}^{-1} \cdot (\boldsymbol{l}-\boldsymbol{\mu}_{z_i})\right). \tag{8}$$

In this equation, the mean vector $\boldsymbol{\mu}_{z_i}$ and the covariance matrix $\boldsymbol{\Sigma}_{z_i}$ specify the signal and the noise associated to the intermediate value $z_i$. It therefore remains to estimate them. As for the estimation of the SNR in Section 2.2.1, we denote as $\boldsymbol{l}(z_i)$ the leakage traces collected by measuring the target implementations for random plaintext and keys so that (only) the intermediate value $z_i$ that we want to profile is fixed. We then compute the empirical means and covariances:

$$\hat{\boldsymbol{\mu}}_{z_i} = \hat{\mathsf{E}}_{\frac{q_{\mathsf{p}}}{256}}\Big(\boldsymbol{L}(z_i)\Big), \tag{9}$$

$$\hat{\boldsymbol{\Sigma}}_{z_i} = \hat{\mathsf{E}}_{\frac{q_{\mathsf{p}}}{256}}\Big(\big(\boldsymbol{L}(z_i)-\hat{\boldsymbol{\mu}}_{z_i}\big)\cdot\big(\boldsymbol{L}(z_i)-\hat{\boldsymbol{\mu}}_{z_i}\big)^{\mathsf{T}}\Big). \tag{10}$$

**Extraction.** Suppose the evaluator is provided with $q_{\mathsf{a}}$ (fresh) attack traces. For any trace $\boldsymbol{l}^{\iota}$, with $1 \leqslant \iota \leqslant q_{\mathsf{a}}$, and key byte candidate $k_i^*$, she can use the previously estimated model and compute $\hat{\mathsf{f}}(\boldsymbol{l}^{\iota}|x_i^{\iota}, k_i^*) = \mathsf{f}\big(\boldsymbol{l}^{\iota}|\hat{\boldsymbol{\mu}}_{z_i}, \hat{\boldsymbol{\Sigma}}_{z_i}\big)\big)$. In a maximum

likelihood attack, the evaluator then selects her key byte candidate as:

$$\check{k}_i = \underset{k_i^*}{\mathsf{argmax}}\ \hat{\mathsf{f}}\big(\bar{\boldsymbol{l}}|\boldsymbol{x}_i, k_i^*\big) = \underset{k_i^*}{\mathsf{argmax}} \prod_{\iota=1}^{q_\mathsf{a}} \hat{\mathsf{f}}\big(\boldsymbol{l}^\iota|x_i^\iota, k_i^*\big), \tag{11}$$

where we recall that the notations $\bar{\boldsymbol{l}}$ and $\boldsymbol{x}_i$ are for the $q_\mathsf{a} \times \tau$ leakage matrix and $q_\mathsf{a}$-element plaintext byte vector. In a maximum a posteriori attack, she additionally applies Bayes' law and selects her key byte candidate as:

$$\check{k}_i = \underset{k_i^*}{\mathsf{argmax}}\ \hat{\mathsf{Pr}}\big(k_i^*|\boldsymbol{x}_i, \bar{\boldsymbol{l}}\big) = \underset{k_i^*}{\mathsf{argmax}}\ \frac{\hat{\mathsf{f}}\big(\bar{\boldsymbol{l}}|\boldsymbol{x}_i, k_i^*\big) \cdot \mathsf{Pr}(k_i^*)}{\sum_{k_i'=0}^{255} \hat{\mathsf{f}}\big(\bar{\boldsymbol{l}}|\boldsymbol{x}_i, k_i'\big) \cdot \mathsf{Pr}(k_i')}, \tag{12}$$

where we use the shortcut notation $\mathsf{Pr}(X = x) = \mathsf{Pr}(x)$ for conciseness. The attack data complexity of both approaches is equivalent in the (frequent) case of uniformly distributed keys. However, they slightly differ when it comes to exploiting their outcomes for key enumeration, as discussed in Section 2.5.

Note that in the original TA, the evaluator estimates one covariance matrix per target intermediate value. But under the (homoscedasticity) assumption that the noise distribution is the same for all values, it is possible to compute a single so-called pooled covariance matrix [CK13], which is obtained by averaging the covariance matrices of all values. If this assumption is correct, it provides a better estimate since it uses 256 times more traces (in our AES S-box case study).

Theoretically, templates can be computed for the raw leakage traces. Yet, such a direct approach can have a high profiling complexity when leakage traces are long. Concretely, it is therefore usually applied on reduced traces that contain the informative POIs. If the target intermediate computation to profile leads to a small number of informative time samples, directly profiling Gaussian templates for all the POIs can be possible. If not, selecting the POIs to retain is a harder problem, since the best tuple of POIs may not be the set of samples with the highest (univariate) detection metric. Dimensionality reduction tools can then be used to exploit multivariate information in a more principled manner.

(*Exercises 1.4 & 1.5 are suggested to further illustrate the previous content.*)

### 2.3.2 Dimensionality reduction

Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are two standard techniques to reduce the dimensionality of observations to classify.

In the context of side-channel analysis, they both aim to project the leakage traces in a subspace where either the side-channel signal (in the case of PCA [APSQ06]) or the side-channel SNR (in the case of LDA [SA08]) is maximized.

**PCA-proj.** Looking at the definition of SNR in Section 2.2.1, the side-channel signal corresponds to the (inter-class) variance between the empirical mean traces $\{\hat{\boldsymbol{\mu}}_{z_i}\}_{256}$. PCA therefore looks for the first $N_c$ principal directions $\{\hat{\boldsymbol{w}}_c\}_{N_c}$ such that $N_c \leqslant \tau$ which form an orthonormal basis of the $N_c$-dimensional subspace capturing most of the the variance of $\{\hat{\boldsymbol{\mu}}_{z_i}\}_{256}$. It is shown in [Jol86] that these principal directions are the eigenvectors $\hat{\boldsymbol{U}}$ of the signal covariance matrix:

$$\hat{\boldsymbol{\Sigma}}_{\mathsf{s}} = \frac{1}{256} \cdot \sum_{z_i=0}^{255} (\hat{\boldsymbol{\mu}}_{z_i} - \bar{\boldsymbol{\mu}}) \cdot (\hat{\boldsymbol{\mu}}_{z_i} - \bar{\boldsymbol{\mu}})^{\mathsf{T}}, \tag{13}$$

with $\hat{\boldsymbol{\Sigma}}_{\mathsf{s}} = \hat{\boldsymbol{U}}\hat{\boldsymbol{\Delta}}\hat{\boldsymbol{U}}$ and where $\bar{\boldsymbol{\mu}} = \frac{1}{256}\sum_{z_i=0}^{255} \hat{\boldsymbol{\mu}}_{z_i}$ is the average of the mean traces. The principal directions $\{\hat{\boldsymbol{w}}_c\}_{N_c}$ are the columns of $\hat{\boldsymbol{U}}$ corresponding to the $N_c$ largest eigenvalues of $\hat{\boldsymbol{\Delta}}$, which we next denote as $\hat{\boldsymbol{W}} \in \mathbb{R}^{\tau \times N_c}$.

The modeling part of a PCA-based TA then requires to estimate the empirical mean and covariance of the traces projected along the principal directions:

$$\hat{\boldsymbol{\nu}}_{z_i} = \hat{\mathsf{E}}_{\frac{q_{\mathsf{P}}}{256}}\Big(\hat{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{L}(z_i)\Big), \tag{14}$$

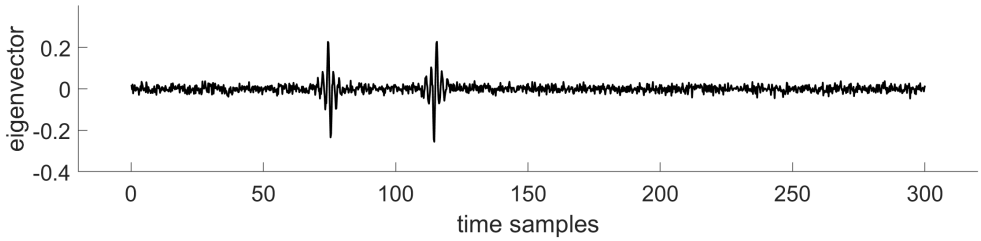
$$\hat{\boldsymbol{\Lambda}}_{z_i} = \hat{\mathsf{E}}_{\frac{q_{\mathsf{P}}}{256}}\Big(\big(\hat{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{L}(z_i) - \hat{\boldsymbol{\nu}}_{z_i}\big) \cdot \big(\hat{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{L}(z_i) - \hat{\boldsymbol{\nu}}_{z_i}\big)^{\mathsf{T}}\Big), \tag{15}$$

where the reduced profiling data complexity pops up if $N_c \ll \tau$.* And in order extract information from a fresh trace $\boldsymbol{l}^{\iota}$, the evaluator can follow the same process as for a TA but with projected traces. For this purpose, she just has to replace the computation of the likelihood $\mathsf{f}\big(\boldsymbol{l}^{\iota}|\hat{\boldsymbol{\mu}}_{z_i}, \hat{\boldsymbol{\Sigma}}_{z_i}\big)$ by $\mathsf{f}\big(\hat{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{l}^{\iota}|\hat{\boldsymbol{\nu}}_{z_i}, \hat{\boldsymbol{\Lambda}}_{z_i}\big)$.

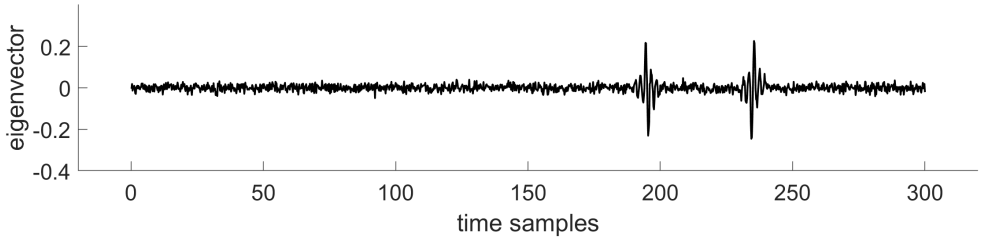
For illustration, the principal direction corresponding to the AES S-boxes' outputs computed with the same simulated traces as previously are given in Figure 9. As expected, the samples selected by PCA match the previously detected ones.

**LDA-proj.** Instead of seeking the directions maximizing the variance of the mean traces, one can look for the directions $\{\tilde{\boldsymbol{w}}_c\}_{N_c}$ maximizing the ratio between the

* If the set of profiling traces is too small to represent the distribution of the leakage traces accurately, it is advisable to split it in two and to estimate the principal directions and the templates with different sets of traces in order to avoid overfitting with the profiling data.




(a) First S-box.




(b) Second S-box.

Figure 9: Principal PCA direction for the AES S-boxes' outputs ($q_{\mathsf{p}} = 1024$).

inter-class scatter $\tilde{\mathbf{\Sigma}}_{\mathsf{S}}$ and the intra-class scatter $\tilde{\mathbf{\Sigma}}_{\mathsf{R}}$ (i.e., the side-channel SNR) after projection. That is, to maximize the ratio $\frac{\tilde{\boldsymbol{w}}^{\mathsf{T}}\tilde{\mathbf{\Sigma}}_{\mathsf{S}}\tilde{\boldsymbol{w}}}{\tilde{\boldsymbol{w}}^{\mathsf{T}}\tilde{\mathbf{\Sigma}}_{\mathsf{R}}\tilde{\boldsymbol{w}}}$ where:

$$\tilde{\mathbf{\Sigma}}_{\mathsf{S}} = \frac{q_{\mathsf{p}}}{256} \cdot \sum_{z_i=0}^{255} (\hat{\boldsymbol{\mu}}_{z_i} - \bar{\boldsymbol{\mu}}) \cdot (\hat{\boldsymbol{\mu}}_{z_i} - \bar{\boldsymbol{\mu}})^{\mathsf{T}}, \tag{16}$$

$$\tilde{\mathbf{\Sigma}}_{\mathsf{R}} = \sum_{z_i=0}^{255} \sum_{j=1}^{\frac{q_{\mathsf{p}}}{256}} (\boldsymbol{l}^j - \hat{\boldsymbol{\mu}}_{z_i}) \cdot (\boldsymbol{l}^j - \hat{\boldsymbol{\mu}}_{z_i})^{\mathsf{T}}. \tag{17}$$

Note that these quantities are equal to covariances up to multiplicative constants that do not play a role in the next derivation (hence the slight abuse of notation). Since $\tilde{\mathbf{\Sigma}}_{\mathsf{S}}$ is positive definite and symmetric and since $\tilde{\boldsymbol{w}}$ is scale-invariant, the maximization problem can be replaced by the following eigendecomposition:

$$\tilde{\mathbf{\Sigma}}_{\mathsf{S}}^{\frac{1}{2}} \tilde{\mathbf{\Sigma}}_{\mathsf{R}}^{-1} \tilde{\mathbf{\Sigma}}_{\mathsf{S}}^{\frac{1}{2}} = \tilde{\boldsymbol{U}} \tilde{\mathbf{\Delta}} \tilde{\boldsymbol{U}}^{\mathsf{T}}, \tag{18}$$

where $\tilde{\mathbf{\Sigma}}_{\mathsf{S}} = \tilde{\boldsymbol{U}}_{\mathsf{S}} \tilde{\mathbf{\Delta}}_{\mathsf{S}} \tilde{\boldsymbol{U}}_{\mathsf{S}}^{\mathsf{T}}$ implies $\tilde{\mathbf{\Sigma}}_{\mathsf{S}}^{\frac{1}{2}} = \tilde{\boldsymbol{U}}_{\mathsf{S}} \tilde{\mathbf{\Delta}}_{\mathsf{S}}^{\frac{1}{2}} \tilde{\boldsymbol{U}}_{\mathsf{S}}^{\mathsf{T}}$ and the projection directions are given by $\tilde{V} = \tilde{\mathbf{\Sigma}}_{\mathsf{S}}^{\frac{-1}{2}} \tilde{\boldsymbol{U}}$. As in PCA-based TAs, we then stack the directions

corresponding to the $N_c$ largest eigenvalues of $\tilde{\mathbf{\Delta}}$ in a projection matrix $\tilde{\boldsymbol{W}} \in \mathbb{R}^{\tau \times N_c}$ and perform the modeling and information extraction using $\tilde{\boldsymbol{W}}$.

*Terminology.* The exposition of PCA and LDA in this section follows their introduction in the side-channel literature, where they are described as dimensionality reduction tools to be used in complement to TAs [APSQ06, SA08]. For PCA this is natural since it is indeed generally viewed as projection. For LDA, this may lead to a confusion since the same term is also used in the machine learning literature to directly denote a linear classifier [Bis07]. In order to avoid this confusion, we will next denote the use of PCA and LDA for dimensionality reduction as PCA-proj. and LDA-proj. It can be shown that the combination of LDA-proj. with a pooled TA is equivalent to the LDA classifier, while the combination of LDA-proj. with a TA rather corresponds to a Quadratic Discrimant Analysis (QDA) classifier.

### 2.3.3 Linear regression

Besides the size of the leakage traces that can make the estimation of the corresponding covariance matrix complex, the other parameter that increases the cost of profiling is the number of templates to build. In the simple AES case study we analyze in this chapter, this is not a critical issue since estimating 256 mean vectors and covariance matrices is easy. But in more advanced cases where (for example) 32-bit intermediate computations are considered, estimating $2^{32}$ templates may become a bottleneck. Originally introduced in the context of side-channel analysis by Schindler et al. [SLP05], Linear Regression (LR) has been rapidly established as an efficient alternative to TA in this context. The main idea of LR-based attacks is to approximate the deterministic part of the leakage function (defined in Equation 2) as a weighted sum of $N_b$ well-chosen basis functions:

$$\delta_j(z_i) \approx \sum_{\kappa=1}^{N_b} a_j^{\kappa} \cdot \mathsf{b}_j^{\kappa}(z_i), \tag{19}$$

with the $a_j^{\kappa}$'s $\in \mathbb{R}$. For an 8-bit value $z_i$, a typical choice is to consider $N_b = 9$ and use the bits of $z_i$ with an additional constant term as basis functions. The approximation can then be refined by adding more basis functions. For example, we can add the $\binom{8}{d}$ basis functions of degree $d$, which we denote as quadratic leakage models if $d \leqslant 2$, cubic leakage models if $d \leqslant 3$, … The number of basis functions can increase up to the case where the monomials of all degrees (up to 8 in our example) are used, leading to $N_b = 256$ and $\delta_j(z_i) = \hat{\mu}_{z_i}$, as in a TA. Here, the reduced profiling data complexity pops up if there are good leakage models

with $N_b < 256$. The latter is expected to be observed for CMOS implementations, where the power consumption or electromagnetic radiation depends on the number of bit switches weighted by the bits' load capacitances [RCN02].

Profiling the model of an LR-based attack first requires to estimate the $N_b$-element vector of coefficients $\boldsymbol{a}_j$ for each time sample $j$, by applying the least square method. For this purpose, the evaluator collects a matrix of $\frac{q_\mathsf{p}}{2}$ profiling traces, next denoted as $\bar{\boldsymbol{l}} = [\boldsymbol{l}^1, \boldsymbol{l}^2, \ldots, \boldsymbol{l}^{\frac{q_\mathsf{p}}{2}}]$, where the elements correspond to target intermediate values $z_i^1, z_i^2, \ldots, z_i^{\frac{q_\mathsf{p}}{2}}$. She then builds a $\frac{q_\mathsf{p}}{2} \times N_b$ matrix:

$$\bar{\boldsymbol{B}}_j = \begin{pmatrix} \mathsf{b}_j^1(z_i^1) & \mathsf{b}_j^2(z_i^1) & \cdots & \mathsf{b}_j^{N_b}(z_i^1) \\ \mathsf{b}_j^1(z_i^2) & \mathsf{b}_j^2(z_i^2) & \cdots & \mathsf{b}_j^{N_b}(z_i^2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{b}_j^1(z_i^{\frac{q_\mathsf{p}}{2}}) & \mathsf{b}_j^2(z_i^{\frac{q_\mathsf{p}}{2}}) & \cdots & \mathsf{b}_j^{N_b}(z_i^{\frac{q_\mathsf{p}}{2}}) \end{pmatrix},$$

and the vector of coefficients minimizing the model mean square error is:

$$\hat{\boldsymbol{a}}_j = [\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{N_b}] = (\bar{\boldsymbol{B}}_j^\mathsf{T} \bar{\boldsymbol{B}}_j)^{-1} \bar{\boldsymbol{B}}_j^\mathsf{T} \boldsymbol{l}_j, \ \forall j \in [1:\tau], \tag{20}$$

where $\boldsymbol{l}_j$ denotes the vector containing the $j$th time sample $l_j^\iota$ for each leakage trace $\boldsymbol{l}^\iota$ of $\bar{\boldsymbol{l}}$. Given an estimation of the deterministic part of the leakage function, the evaluator then uses another set of $\frac{q_\mathsf{p}}{2}$ profiling traces in order to estimate its probabilistic part. For this purpose, LR-based attacks rely on the same multivariate Gaussian assumption as TA. As a result, the evaluator just has to compute $\frac{q_\mathsf{p}}{2}$ noise traces corresponding to various $z_i$ values, which we denote as:

$$\boldsymbol{r}^\iota = [r_1^\iota, r_2^\iota, \ldots, r_\tau^\iota] = [l_1^\iota - \hat{\delta}_1(z_i^\iota), l_2^\iota - \hat{\delta}_2(z_i^\iota), \ldots, l_\tau^\iota - \hat{\delta}_\tau(z_i^\iota)], \tag{21}$$

and to estimate a single covariance matrix $\hat{\boldsymbol{\Sigma}}$ by pooling all the $\boldsymbol{r}^\iota$ traces. Eventually, the information extraction part of a LR-based attack again proceeds like a TA, where the likelihoods of the fresh leakage traces $\mathsf{f}\big(\boldsymbol{l}^\iota | \hat{\boldsymbol{\mu}}_{z_i}, \hat{\boldsymbol{\Sigma}}_{z_i}\big)\big)$ are replaced by the likelihoods of fresh noise traces $\mathsf{f}\big(\boldsymbol{r}^\iota | \hat{\boldsymbol{\mu}}_{z_i}, \hat{\boldsymbol{\Sigma}}\big)\big)$ and the mean vectors are directly given by the deterministic approximation $\boldsymbol{\mu}_{z_i} = \hat{\boldsymbol{\delta}}(z_i)$ with:

$$\hat{\boldsymbol{\delta}}(z_i) = [\hat{\delta}_1(z_i), \hat{\delta}_2(z_i), \ldots, \hat{\delta}_\tau(z_i)]. \tag{22}$$

Note that the balance between the number of traces needed to estimate the deterministic part and the probabilistic part of the LR-based models can be adapted in function of the number of basis function (since large $N_b$ values make the estimation of the deterministic part more complex) and the level of noise and size of the traces (which determine the estimation cost of the probabilistic part).

### 2.3.4 Discussion and related works

*Which tool to use?* Information extraction tools can be evaluated along two axes: their profiling complexity and the complexity of an attack using their estimated model. As a first step towards discussing which information extraction tool to privilege in a worst-case security evaluation, Figure 10 illustrates (5 independent) attack results for the different tools described in this section. Again, these analyses are performed against the same case study as in Chapter 1. For the profiled analyses, we further made sure that the profiling data was sufficient to estimate the model parameters accurately. It leads to the following observations.

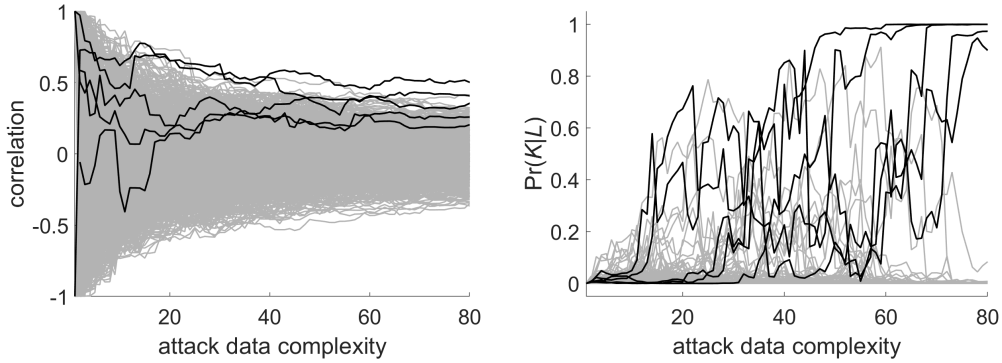
First, and for each information extraction tool, the five attacks do not lead to the same outcome. Such probabilistic results are expected due to the probabilistic nature of side-channel analyses. But it also means that comparing the attack outcomes requires a metric that can cope with random experiments. This issue of side-channel evaluation metrics will be specifically discussed in Chapter 3.

Second, and looking at general trends, the figure highlights gradually improving attack efficiencies when decreasing the amount of leakage assumptions and increasing the number of POIs (aka dimensions) exploited in the attacks.

The gain obtained when getting rid of leakage assumptions is obvious when comparing the unprofiled CPA of Figure 10a with the profiled CPA of Figure 10c or the univariate TA of Figure 10d. These figures confirm the observation already made in Section 2.2.2 that the leakage traces used in these attacks do not exactly follow a Hamming weight model. Figure 10b then shows an interesting intermediate situation where a LR-based attack using a 9-element (linear) basis improves over the unprofiled CPA while not reaching the data complexity of the univariate TA.* Next, the gain obtained when increasing the number of POIs exploited in the attacks appears when comparing the univariate TA of Figure 10d with the bivariate TA of Figure 10e and the PCA-based TA reported in Figure 10f.

On the other hand, using less leakage assumptions and modeling the leakage of more POIs also has an impact on the profiling complexity of the evaluations. Again, a quantitative comparison will require sound evaluation metrics, of which the definition and discussion is deferred to Chapter 3. Meanwhile, an informal illustration of the high-level intuitions behind our different information extraction tools is given in Table 2. The first and third lines show the straightforward tradeoff between an unprofiled CPA and a profiled TA. The former assumes a Hamming

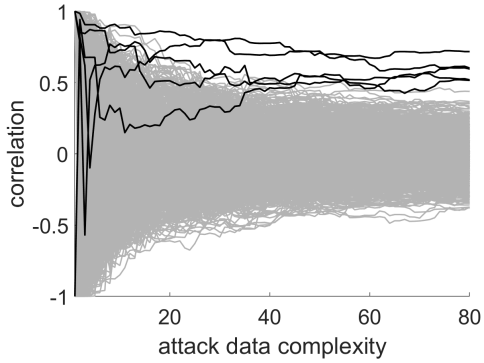
* The leakage function in our simulations was chosen such that the correlation between its noise-free outputs and a well-estimated regression-based model with a linear basis equals 0.75.



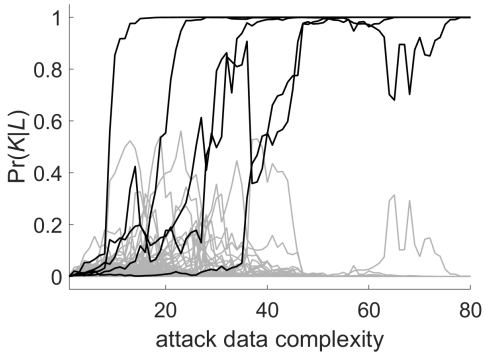

(a) Unprofiled CPA.



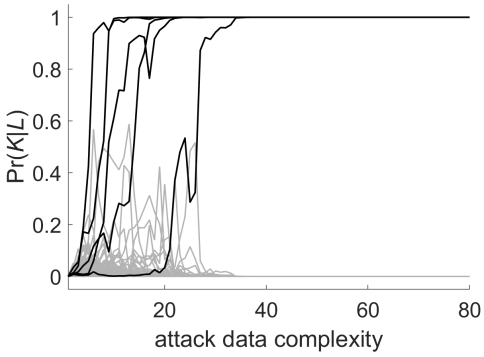

(b) Univatiate LR ($N_b = 9$).



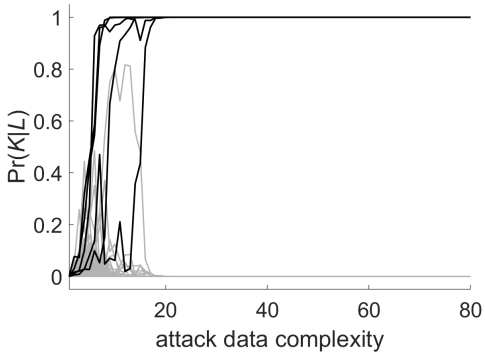

(c) Profiled CPA.




(d) Univariate TA.




(e) Bivariate TA.




(f) PCA-based TA ($N_c = 1$).

Figure 10: Information extraction for the first AES key byte. Black curves are for the good keys of 5 independent attacks. Grey curves are for wrong key candidates.

weight leakage model and is inherently univariate. So it falls short if the actual leakage distribution significantly deviates from the Hamming weight assumption and it is unable to exploit multiple POIs jointly. The latter does not make any assumption on the deterministic part of the leakage function and can be extended to analyze multivariate distributions. So it leads to reduced attack complexities. But this comes at the cost of a possibly expensive profiling phase where many templates must be built, which requires the estimation of a possibly large covariance matrix (in case pooled TA are considered) or many such matrices (if the homoscedasticity assumption does not hold, which prevents pooling).

| | ***assumptions*** | | ***#POIs*** | ***profiling data complexity*** | | ***flavor*** |
|---|---|---|---|---|---|---|
| **extraction tool** | $\delta$ | ***noise*** | ***exploited*** | ***# templates*** | ***size of*** $\Sigma$ | |
| unprofiled CPA | HW | Gaussian + homoscedasticity | one | $\emptyset$ | $\emptyset$ | metric |
| profiled CPA | $\emptyset$ | Gaussian + homoscedasticity | one | $2^\eta$ | $\emptyset$ | metric |
| TA | $\emptyset$ | Gaussian opt. homoscedasticity | many | $2^\eta$ | $\tau \times \tau$ | probabilistic |
| PCA-proj. LDA-proj. +TA | $\emptyset$ | Gaussian opt. homoscedasticity | many | $2^\eta$ | $\propto N_c$ | probabilistic |
| LR-based | basis functions | Gaussian + homoscedasticity | many | $\propto N_b$ | $\tau \times \tau$ | probabilistic |

Table 2: Informal classification of information extraction tools for an $\eta$-bit target.

The bottom lines of Table 2 then show that dimensionality reduction and LR-based attacks can be used to reduce the profiling data complexity. PCA-proj. and LDA-proj. work by reducing the size of the covariance matrices to estimate as a function of the number of dimensions retained after projection $N_c$. LR-based attacks work by approximating the deterministic part of the leakage function with $N_b$ well-chosen basis functions, which reduces the number of parameters to estimate. When possible, combining dimensionality reduction with a TA is a good starting point: it allows to efficiently extract information from multiple POIs and only relies on a Gaussian assumption that is usually acceptable in the unprotected implementation context that we consider in this first part of the book.

The table additionally highlights that distinguishers may have a metric or probabilistic flavor, which is reflected by the very different shapes of the CPA plots compared to the other plots in Figure 10. By metric flavor, we mean that the

asymptotic value of the CPA distinguisher (when $q_a$ increases) can be interpreted as an indication of the security level, which we will formalized in Section 3.2.1. By probabilistic flavor, we mean that the other distinguishers output probabilities. As a result, their asymptotic value (which is always one for successful attacks) does not give any hint on the security level. By contrast, they directly give an indication whether the corresponding key byte has been recovered with confidence, which we will show is convenient for optimal key enumeration in Section 2.5.

*Implementation issues.* As discussed in Section 2.3.2, LDA optimizes a more relevant criteria than PCA at the cost of estimating an inter-class and an intra-class scatter. Directly computing these matrices on raw traces can be expensive. As a result, leveraging a detection step in order to identify the POIs leading to a sufficient side-channel SNR is usually a good approach. In the context of unprotected implementations, it is expected that computing LDA projections on POIs will lead to acceptable profiling complexities. If not, PCA can serve as an alternative solution since it benefits from a "small dataset" variant: for an 8-bit target intermediate value, it allows computing the first 255 principal directions by estimating a $256 \times 256$ matrix rather than the $\tau \times \tau$ covariance matrix [APSQ06].

**Related works.** The application of TA and LR-based attacks can be non-trivial and concrete guidelines for their efficient implementation are respectively discussed in [CK13] and [LPR13]. Profiled side-channel analysis also faces the issue that the models estimated on a profiling device may not exactly match the attack device [RSV$^+$11, EG12, CK14]. Robust profiling has been introduced in order to mitigate such concerns [WO15, CK18]. Various alternatives and improvements of the PCA and LDA projections have been proposed in the literature, e.g., [DSV$^+$15, CDP15]. Besides the profiled analyses we consider in this chapter, non-profiled attacks have also been a topic of intense research. Popular examples include Mutual Information Analysis (MIA) [GBTP08] and its follow ups [PR09, VS09, WOM11], the non-profiled extension of LR-based attacks [DPRS11] and its follow ups [WOS14, WYS$^+$18], and many others [LCC$^+$06, BGL08, BGL09], some of them compared in [SGV08]. Leaving the worst-case evaluation setting and assuming an evaluator with limited understanding of her target implementation raises the question whether the models estimated during profiling completely capture the target leakage, which can be tested with the appropriate tools as well [GO22]. Besides, while most side-channel attacks extract direct information on the target intermediate computations of a leaking device, collision attacks rather try to detect whether some intermediate variables are identical, which can also be carried out thanks to profiled and non-profiled distinguishers [SLFP04, LMV04, Bog07a, MME10, GS13, GG19]. Eventually, the TA and LR-based attacks in this section rely on a Gaussian assumption, which is usually acceptable for unprotected implementations. It is however possible to consider other (higher-order,

mixture) distributions (e.g., [LP07, MS16, SMSG16]) or to model them with advanced machine learning learning tools (e.g., [HGM+11, HZ12, LBM14, MPP16]. We defer their discussion to Part III of the book where we consider protected implementations.

## 2.4 Information exploitation

A trivial solution to exploit side-channel information, already outlined in Chapter 1, is to target independent operations such as the sixteen S-boxes of the first (or last) AES round independently. Yet, an obvious limitation of this approach is that it does not allow exploiting the leakage of internal rounds, due to computational reasons. This is because divide-and-conquer attacks rely on guessing the part of the key on which their target sensitive variable depends. For the S-box layer in the first (or last) AES round, this corresponds to a low time complexity of $16 \times 2^8$. If additionally targeting the MixColumn operation, it grows to $4 \times 2^{32}$ which is tractable but computationally intensive [MOW14]. Extending this strategy beyond the first round (i.e., once diffusion is complete) is not possible. In this section, we therefore focus on so-called analytical attacks, which we define in contrast to divide-and-conquer ones as attacks able to efficiently exploit the leakage of operations that depend on more key bits than can be enumerated (e.g., after diffusion is complete in a block cipher). For this purpose, we focus on the Soft Analytical Side-Channel Attacks (SASCA) introduced in [VGS14].

### 2.4.1 Soft Analytical Side-Channel Attacks

The high-level idea of a SASCA is to represent a leaking implementation as a factor graph, with two types of nodes and bidirectional edges. So-called variable nodes represent the intermediate variables. So-called factor nodes represent the a-priori knowledge about the variables (e.g., the known plaintexts and the side-channel information) and the operations applied to these variables. The nodes are then connected via bidirectional edges that are used to propagate the (local) information collected about every leaking variable through the graph. Concretely, the proposal of [VGS14] is to use Belief Propagation (BP) for this purpose [Pea82], which is a message-passing algorithm for performing inference on graphical models. BP has guaranteed convergence when applied to trees and only provides heuristic results when applied to general graphs with cycles or loops.

We next describe the functioning of a SASCA by applying it to the execution of two AES S-boxes of which the output is then XORed, as happens when executing the beginning of the MixColumns operation. It is based on three steps:

The first step is the *construction of a factor graph*, which we illustrate in Figure 11. It represents the same computations $z_0 = \mathsf{S}(x_0 \oplus k_0)$ and $z_1 = \mathsf{S}(x_1 \oplus k_1)$ as in Chapter 1, with an additional operation $v = z_0 \oplus z_1$ (that takes place in MixColumns). It is important to remark that the sensitive variables $y_i$ and $z_i$ are merged in the graph. This is because they are bijectively connected and a single template will extract information on both of them. Factor nodes are denoted as $\mathcal{P}$ for known plaintexts, $\mathcal{L}$ for leakage information and $\oplus$ for the XORs.

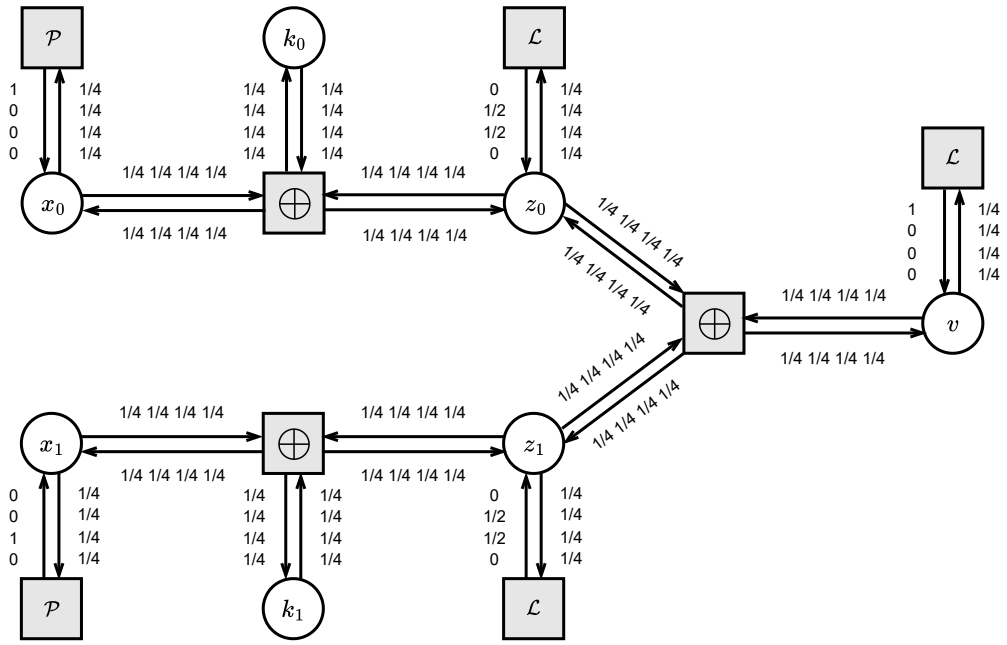



Figure 11: Exemplary factor graph and initialization of a SASCA.

The second step is the *information extraction*, which allows initializing the two types of messages in the factor graph. Each of them correspond to lists of probabilities which, for simplicity, we illustrate with a 2-bit example in Figure 11. The first ones go from variables (circles) to factors (squares) and are initialized with a uniform distribution. The second ones go from factors (squares) to variables (circles) and are initialized with a uniform distribution for the XOR factor nodes,

with probabilities inferred using one of the methods described in Section 2.3 for the $\mathcal{L}$ factor nodes and with a one-hot vector for the $\mathcal{P}$ factor nodes.

The last step is a *decoding* one, where the BP algorithm is applied. It works by iteratively updating the two types of messages based on two rules. The first one updates the variable-to-factor messages $\underset{v \to f}{m}$ as follows:

$$\underset{v \to f}{m}(\alpha) = \prod_{f' \in \text{neighbors(v)} \setminus f} \underset{f' \to v}{m}(\alpha), \tag{23}$$

where $\alpha$ is the value taken by the target variable. For example, on the top of Figure 12, the plaintext information on $x_0$ is directly forwarded to the variable-to-factor message going from $x_0$ to the XOR, and the leakage on $z_0$ is forwarded to the variable-to-factor messages going from $z_0$ to its two neighboring XORs as the product $[0, 1/2, 1/2, 0] \circ [1/4, 1/4, 1/4, 1/4] = [0, 1/8, 1/8, 0]$, normalized to sum at one on the figure, and where $\circ$ is the element-wise Hadamard product.

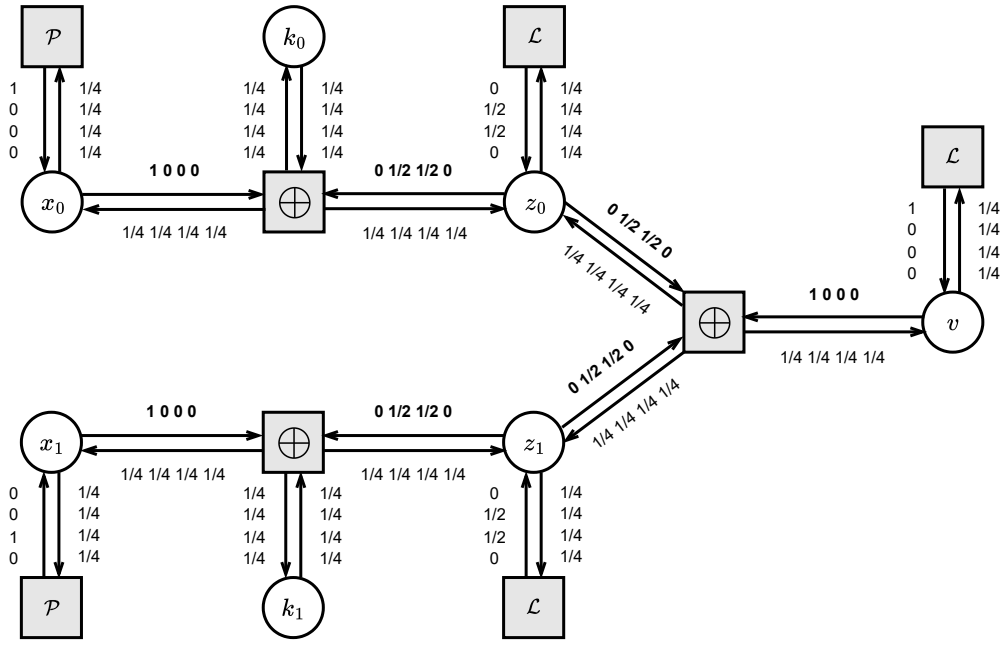



Figure 12: Update of the variable-to-factor messages.

The second rule updates the factor-to-variable messages as follows:

$$\underset{f \to v}{m}(\alpha) = \sum_{\alpha' \in \text{ inputs of neighbors(f)} \setminus v} \text{f}(\alpha, \alpha') \cdot \prod_{v' \in \text{ neighbors(f)} \setminus v} \underset{v' \to f}{m}(\alpha'), \qquad (24)$$

where $\text{f}(\alpha, \alpha')$ is a function that outputs one if the XOR between the inputs of the neighbors of target factor node excluding the target variable node is correct.

Let us illustrate this second rule with the update of the factor-to-variable message going from the middle XOR to $v$ based on the variable-to-factor messages going from $z_0$ and $z_1$ to this XOR, depicted in Figure 13. Starting with the case where $\alpha = 0$, we can see that $\text{f}(\alpha, \alpha') = 1$ if $\alpha' = (0, 0)$, which leads to a product of variable-to-factor messages $0$ (i.e., $\text{f}(\alpha, \alpha') = 1$ since $0 \oplus 0 = 0$ and the product of the first elements of the variable-to-factor messages equals 0), $\alpha' = (1, 1)$, which leads to a product of variable-to-factor messages 1/4, $\alpha' = (2, 2)$, which leads to a product of variable-to-factor messages 1/4 and $\alpha' = (3, 3)$ which leads to a product of variable-to-factor messages 0. The sum of these products is 1/2 (i.e., the first element of the factor-to-variable that we update on the figure). Similarly, we have a sum of 0 for $\alpha = 1$ and $\alpha = 2$ and a sum of 1/2 for $\alpha = 3$.

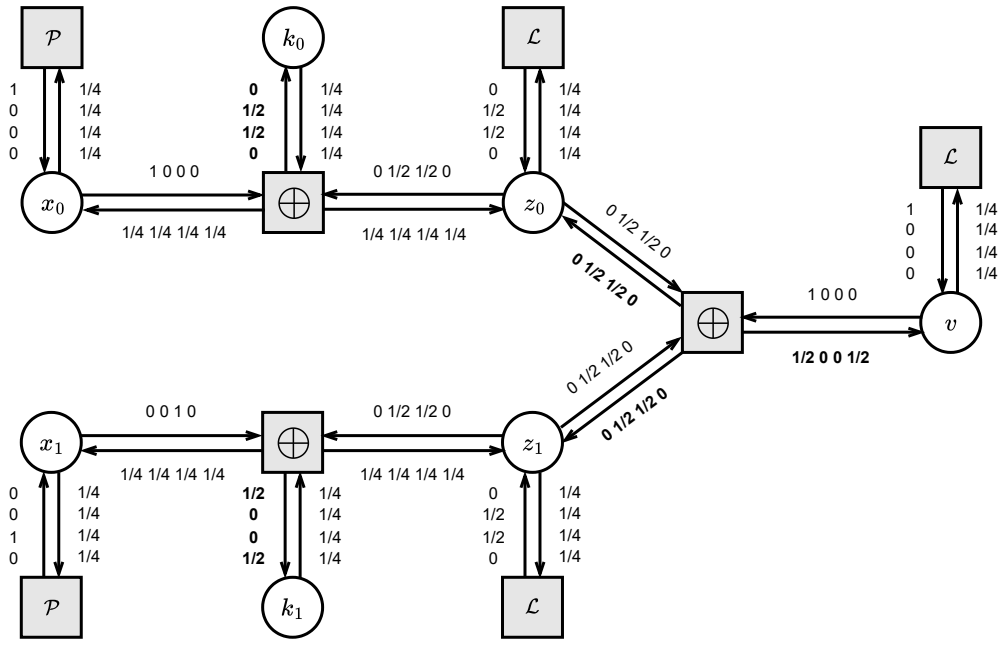



Figure 13: Update of the factor-to-variable messages.

One iteration of BP consists in applying these two rules to each variable-to-factor and factor-to-variable message. If the factor graph is a tree, the algorithm should converge to exact probabilities for its variables after a number of iterations at most equal to the diameter of the graph. If the factor graph contains cycles or loops, convergence is not guaranteed, but BP usually gives good approximated results after a number of iterations slightly larger than the graph diameter.

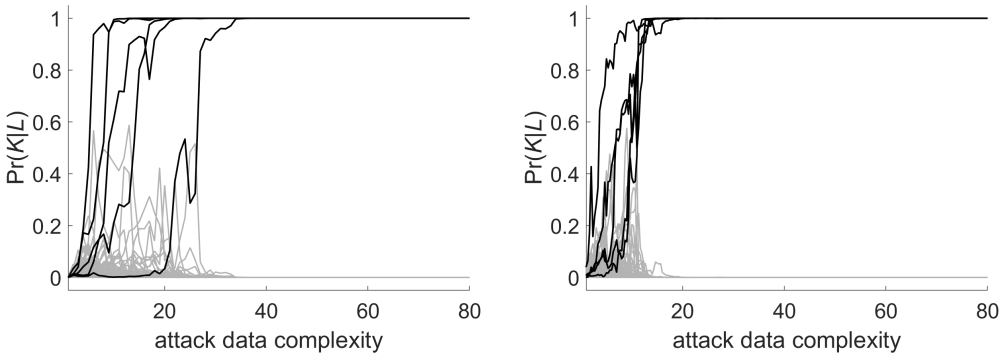
### 2.4.2 Discussion and related works

The time and memory complexity of a SASCA are determined by the size of the target intermediate variables and the size of the factor graph. In particular, in the trivial implementation of BP outlined above, propagating information through the XOR operation has a complexity in $\mathcal{O}(2^{2\eta})$ for $\eta$-bit variables, which is generally the bottleneck. As observed in [KPP20], the factor-to-variable update rule for a computation $v = z_0 \oplus z_1$ can be viewed as a convolution product. Therefore, the distribution of $Z$ can be obtained via two Walsh-Hadamard transforms on $Z_0$ and $Z_1$, a Hadamard product and an inverse transform. That is, denoting the vector of probabilities of $Z_0$, $Z_1$ and $V$ as $\hat{\boldsymbol{p}}_{z_0}$, $\hat{\boldsymbol{p}}_{z_1}$ and $\hat{\boldsymbol{p}}_v$, we have:

$$\hat{\boldsymbol{p}}_v = \hat{\boldsymbol{p}}_{z_0} * \hat{\boldsymbol{p}}_{z_1} = \texttt{iWHT}\Big(\texttt{WHT}(\hat{\boldsymbol{p}}_{z_0}) \circ \texttt{WHT}(\hat{\boldsymbol{p}}_{z_1})\Big), \tag{25}$$

where $*$ is the convolution product, WHT is the Walsh-Hadamard transform and iWHT its inverse, both of which having a complexity in $\mathcal{O}(2^\eta \cdot \eta)$.

The results of 5 independent SASCA corresponding to the factor graph of Figures 11, 12 and 13 generalized to 8-bit variables are illustrated in (the right part of) Figure 14, where the leakage on $z_0$ and $z_1$ is extracted with a bivariate TA while the leakage on $v$ is extracted with a univariate TA. The comparison with a bivariate TA (reproduced from Figure 10 in the left part of Figure 14) highlights the improved data complexity of SASCA when leveraging the same information extraction tool as a divide-and-conquer attack. It naturally questions how much such a trend can be extended by leveraging factor graphs that capture a full target implementation. In general, digging into a cipher's rounds indeed improves the attacks' data complexity, yet with a decreasing impact as the additional target operations get remote from the ones that can be exploited via a divide-and-conquer approach. How much the exploitation of the inner rounds can improve the attacks is typically cipher-dependent. For the AES and as a rule-of-thumb, gains are generally significant until factor graphs capturing two to three rounds [GGSB20].




(a) Bivariate TA. (b) SASCA against $K_0$.

Figure 14: SASCA vs. bivariate TA for 5 independent attacks.

**Related works.** A more formal introduction to BP can be found in MacKay's book [Mac03]. The general goal of exploiting more target intermediate variables than with a divide-and-conquer approach was formulated in different papers before SASCA were introduced. For example, algebraic side-channel collision attacks are able to exploit the leakage of the first two AES rounds [BKP08], while algebraic side-channel attacks can theoretically exploit the leakage of any round [RSV09, CFGR12]. The main limitation of these previous works was their limited tolerance to the noise that inevitably affects side-channel measurements [OKPW10, ORSW12]. This limitation actually served as the main motivation for soft analytical strategies rather than purely algebraic ones. The impact of the factor graph on the data complexity of SASCA is systematically analyzed in [GRO18a]. The efficient application of SASCA to bitslice implementations is detailed in [CDSU23]. SASCA has also been used for single-trace attacks in the asymmetric setting [PPM17, PP19, ADP$^+$20, HSST23]. Divide-and-conquer and analytical attacks are by far the most popular strategies to exploit leakage in symmetric cryptography. Yet, the extend-and-prune strategy also turns out to be useful in some contexts, for example when targeting stream ciphers [CRR02]. Finally, adaptive chosen-message side-channel attacks are yet another option to improve the information exploitation step [VS10], although adaptively selecting fresh plaintexts based on the leakage of past ones also slows down the collection of measurements.

## 2.5 Post-processing

The result of the previous information extraction and exploitation steps typically takes the form of a vector $\hat{\boldsymbol{p}}_{k_i}$, with $i \in [0:15]$, that contains the probabilities of the 16 AES key bytes. If $\max(\hat{\boldsymbol{p}}_{k_i}) = k_i$ for all key bytes, the attack is directly suc-

cessful. If not, evaluators are left with the question whether the partial information collected about the key bytes can lead to concrete attacks. A direct approach to answer this question is to launch a key enumeration algorithm [VGRS12]. It allows recombining the vectors of probabilities in order to list full keys from the most likely to the least likely one and to test them (e.g., with a plaintext/ciphertext pair). Quite naturally, this approach is bounded by the enumeration power of the evaluator and gets more expensive when the security of the key increases. This may lead to an uncomfortable situation where an evaluator decides to stop enumeration after testing $2^{64}$ keys while she would have succeeded after $2^{65}$.

Rank estimation has been proposed as a remedy to this limitation [VGS13]. It can be viewed as an evaluator's counterpart to key enumeration where the goal is to approximate the position of a key in a list, next denoted as its rank. The rationale behind this tool is that from an evaluation viewpoint, knowing that a key's rank is between $2^{90}$ and $2^{91}$ is sufficiently accurate to conclude about the (in)security of a leaking implementation. It turns out that such an approximation can be obtained very efficiently given that the evaluator knows the correct key (hence, its probability). We next describe a histogram-based rank estimation algorithm from [GGP$^+$15] that is both efficient and conceptually simple.

### 2.5.1 Histogram-based rank estimation

The algorithm first expresses the outcome of the information extraction and exploitation steps described in the previous sections as vectors of log probabilities, that we denote as $\hat{\boldsymbol{lp}}_{k_i} = \log(\hat{\boldsymbol{p}}_{k_i})$. Then it builds histograms for each of these vectors: $\boldsymbol{h}_i = \text{hist}(\hat{\boldsymbol{lp}}_{k_i}, \text{bins})$, where the same set of $n_b$ equally-sized bins must be used for all histograms. The rank estimation in Algorithm 1 works by convoluting histograms. It exploits the property that for two multisets of numbers $\mathcal{S}_1$ and $\mathcal{S}_2$ of which the distribution is described by $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$, the distribution of the numbers in the multiset $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2 = \{x_1 + x_2 \mid x_1 \in \mathcal{S}_1, x_2 \in \mathcal{S}_2\}$ can approximated by the convolution of $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ if they have the same bin size.

An illustration of Algorithm 1 for two AES key bytes is given in Figure 15, where the rank of the key bytes (in the upper and middle parts) and the rank of their concatenation (in the lower part) is the number of keys in the light grey zone. The leftmost light grey bin, denoted as $\text{bins}(\hat{lp}_k)$ is the one where the correct key lies and the number of bins in the convoluted histograms evolves as $n_c \cdot n_b - (n_c - 1)$ with $n_c$ the number of convolutions. In our AES case, $n_c$ is worth 16 since we

**Algorithm 1** Rank estimation.

**Input.** Full key log probability $\hat{lp}_k = \log\left(\hat{\Pr}(k|\boldsymbol{x}, \bar{\boldsymbol{l}})\right)$ and histograms $\boldsymbol{h}_i$.
**Output.** Approximation of the full key rank $\hat{r}_k$.

*Initialization:* $\boldsymbol{h}_{\text{curr}} = \boldsymbol{h}_0$;

*Histograms convolution:*
**for** $i = 1$ to $n_c - 1$ **do**
  $\boldsymbol{h}_{\text{curr}} = \boldsymbol{h}_{\text{curr}} * \boldsymbol{h}_i$;
**end for**
*Rank estimation:*

$$\hat{r}_k \approx \sum_{i=\text{bins}(\hat{lp}_k)}^{n_c \cdot \eta_b - (n_c - 1)} \boldsymbol{h}_{\text{curr}}(i).$$

target the key byte per byte. But a (computationally) more powerful side-channel adversary could target the keys per 32-bit words, leading to $n_c = 4$.

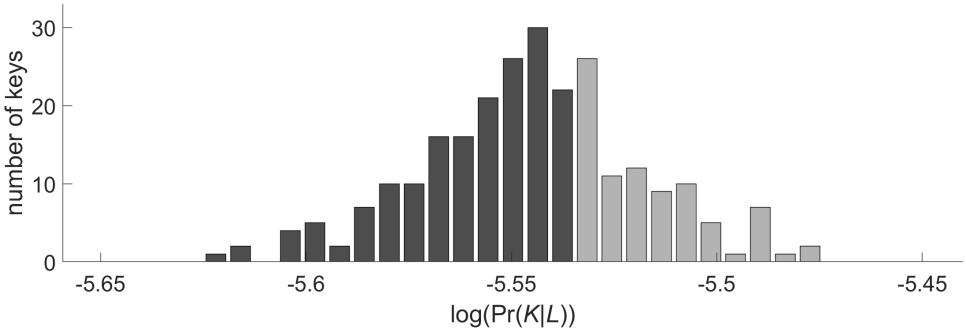
Algorithm 1 only provides an approximation of the key rank, raising the question whether we can bound its errors. For this purpose, we start by explaining the origin of these approximation errors with a 3-bit example. Let us assume that:

$$\begin{aligned} \hat{\boldsymbol{p}}_{k_0} &= [0.22, 0.02, 0.05, 0.12, 0.17, 0.011, 0.2, 0.21], \\ \hat{\boldsymbol{p}}_{k_1} &= [0.07, 0.05, 0.06, 0.20, 0.13, 0.21, 0.06, 0.22], \end{aligned}$$

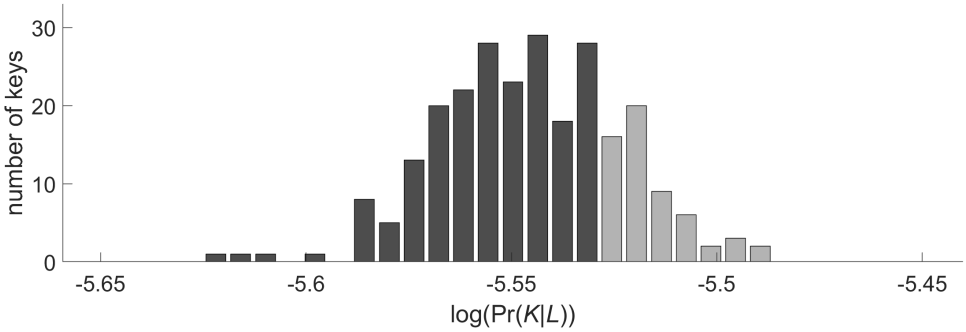
and the correct (concatenated) key is worth $[0, 0]$, so that the first (resp., second) key byte has probability 0.22 (resp., 0.07) after the information extraction and exploitation steps. The corresponding vectors of log probabilities are worth:

$$\begin{aligned} \hat{\boldsymbol{lp}}_{k_0} &\approx [-1.51, -3.96, -3.00, -2.12, -1.77, -4.51, -1.61, -1.56], \\ \hat{\boldsymbol{lp}}_{k_1} &\approx [-2.66, -3.00, -2.81, -1.61, -2.04, -1.56, -2.81, -1.51]. \end{aligned}$$

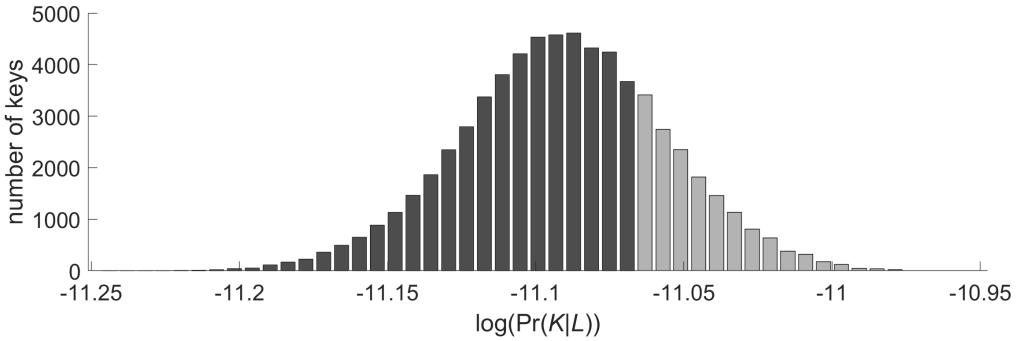
Assuming that $n_b = 3$ bins, it leads to a common bin size of one and histograms with central values $\approx [-4, -3, -2]$ so that we have $\boldsymbol{h}_0 = [2, 1, 5]$ and $\boldsymbol{h}_1 = [0, 4, 4]$. Hence, the convolution $\boldsymbol{h}_0 * \boldsymbol{h}_1$ is a histogram with $n_b = 5$ bins and central values $\approx [-8, -7, -6, -5, -4]$ given by $\boldsymbol{h}_{\text{curr}} = [0, 8, 12, 24, 20]$. The sum of log probabilities $lp_{k_0} + lp_{k_1} = -1.51 - 2.66 = -4.17$, which falls in the rightmost bin. Yet, the central values of the key bytes' bins (before convolution)




(a) Histogram of the log probabilities for $k_0$.




(b) Histogram of the log probabilities for $k_1$.




(c) Histogram of the log probabilities for $k = [k_0, k_1]$.

Figure 15: Histogram-based rank estimation for two AES key bytes.

is -2 and -3. Therefore, the convolution approximates their sum within the bin of central value -5 (rather than -4), leading to an error of one bin.

The previous example suggests that the rank estimation accuracy depends on the quantization errors that can increase each time histograms are convoluted. As a result, a simple bound for the rank estimation can be obtained as:

$$\sum_{i=\mathsf{bins}(\hat{l}p_k)+n_c}^{n_c \cdot n_b-(n_c-1)} \boldsymbol{h}_{\mathsf{curr}}(i) \leqslant \hat{r}_k \leqslant \sum_{i=\mathsf{bins}(\hat{l}p_k)-n_c}^{n_c \cdot n_b-(n_c-1)} \boldsymbol{h}_{\mathsf{curr}}(i). \tag{26}$$

(*Exercise 1.6 is suggested in order to further illustrate the previous content.*)

### 2.5.2 Discussion and related works

For a 128-bit key targeted with 8-bit models, the simple approach outlined in this section already allows reaching sufficiently accurate rank estimations, with a tightness (defined as the logarithm of the ratio between the upper and lower bounds for the key rank) of less than one bit almost instantaneously on a desktop computer. Key enumeration and rank estimation algorithms are sometimes denoted as optimal in the literature. In this respect, it is important to note that it is only the key enumeration and ranking steps of an attack that can be optimal. Namely, assuming that the information extraction and exploitation steps are perfect, optimal key enumeration and rank estimation are not going to add additional errors (besides the ones predicted by the bounds for rank estimation). But on the one hand, the information extraction step relies on statistical models that may be affected by estimation and assumption errors. And on the other hand, the application of SASCA to factor graphs with cycles or loops in inherently heuristic as well. Besides, optimal key enumeration and rank estimation rely on the fact that side-channel distinguishers produce probabilistic outcomes. This is because they use the multiplicative relation $\hat{\mathsf{Pr}}(k_0, k_1|\boldsymbol{x_0}, \boldsymbol{x_1}, \bar{\boldsymbol{l}}) = \hat{\mathsf{Pr}}(k_0|\boldsymbol{x_0}, \bar{\boldsymbol{l}}) \cdot \hat{\mathsf{Pr}}(k_1|\boldsymbol{x_1}, \bar{\boldsymbol{l}})$ to combine the probabilities of two key bytes (and the corresponding additive relation when moving to log probabilities). Such a relation is ensured for the Gaussian TA and regression-based attacks, possibly combined with dimensionality reduction, as described in Section 2.3, but it requires additional heuristics for distinguishers that only output scores like the CPA described in Chapter 1 [CPS16].

**Related works.** Many variants of key enumeration and rank estimation algorithms exist in the literature. Some publications propose solutions to the two problems at once. For

example, a histogram-based approach to key enumeration (that exploits similar ideas as for rank estimation in this section) is described in [PSG16]. Alternatives can be found in [MOOS15, MMO18, Gro18b]. The additional performance gains that may be obtained by relying on a suboptimal enumeration strategy are discussed in [YEM14]. A number of algorithms have also been proposed for rank estimation only [MMOS16, DW17, DW19b, DW19a, DW22, CDS23]. The last one is applicable to analyze dependent keys as for example encoutered when analyzing implementations of the AES-256. Solutions that consider the simpler problem of estimating the average key rank can be found in [CP17, ZDF20, RPC22] – some of them include (heuristic) variants that can operate without knowing the key, a problem that is also discussed in [APSV19]. Finally, some comparisons between different rank estimation methods can be found in [PGS15, YMO22] and the interplay between SACSA and key enumeration is evaluated empirically in [GS15].

# Chapter 3

# Theoretical background

As observed in Section 2.3.4, side-channel analysis is probabilistic in nature, which makes its evaluation challenging. In this chapter, we therefore tackle the problem of sound security assessments with leakage. For this purpose, we first outline a general framework for side-channel key recovery attacks in Section 3.1. It introduces information theoretic and security metrics that can be used to evaluate cryptographic implementations. We also leverage this framework to clarify basic terminology that will help in the rest of the book. Next, in Section 3.2, we show how bounds and shortcuts can contribute to speedup side-channel security assessments and draw connections between the different (types and instances of) metrics introduced. In Section 3.3 we move to the concrete question of how to accurately estimate information theoretic and security metrics, in order to reach reliable conclusions. Eventually, we take a critical view on the tools and metrics introduced in this first part of the book in Section 3.4, where we discuss their optimality and their integration within the worst-case evaluation vision. We conclude by proposing a definition of side-channel attack that will motivate two important types of countermeasures that we investigate in the next parts of the book.

## 3.1 Side-channel analysis framework

Figure 16 depicts the evaluation framework that we use in our investigations. Its main conceptual contribution is a delineation between so-called information theoretic and security metrics. At high level, information theoretic metrics aim to enable the comparison of implementations independent of adversaries. By contrast, security metrics aim to compare how efficiently an adversary can turn side-

channel information into successful attacks. Concretely, this for example implies that information theoretic metrics are independent of the adversary's enumeration power whereas security metrics can be improved by post-processing tools. Depending on the target functionality (e.g., authentication, encryption or authenticated encryption), adversaries can have different goals. As a starting point, we will consider key recovery attacks (i.e., the strongest form of attack) that have been the main focus of concrete side-channel analyses in the literature. We will move to definitions of confidentiality and integrity with leakage in Chapter 4.

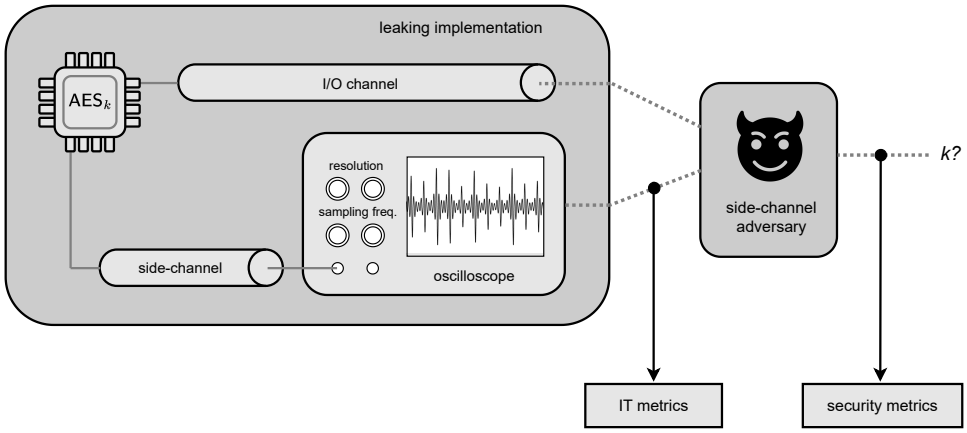



Figure 16: The Eurocrypt 2009 evaluation framework [SMY09].

### 3.1.1 Information theoretic and security metrics

It turns out we already encountered information theoretic metrics. For example, the SNR in Section 2.2.1 and the profiled correlation coefficient in Section 2.2.2 are information theoretic metrics. However, they are limited to the analysis of univariate leakage samples while the most powerful modeling and information extraction tools of Section 2.3 are multivariate. A more general metric proposed for side-channel analysis in [SMY09] is the Mutual Information (MI). It can be used to capture any (multivariate) leakage function. Given a (discrete) target intermediate variable $V$ and a (continuous) leakage vector $\boldsymbol{L}$, the MI is worth:

$$\mathsf{MI}(V; \boldsymbol{L}) = \mathsf{H}(V) + \sum_{v \in \mathcal{V}} \mathsf{Pr}(v) \cdot \int_{\boldsymbol{l} \in \mathcal{L}} \mathsf{f}(\boldsymbol{l}|v) \cdot \log_2 \mathsf{Pr}(v|\boldsymbol{l})\, dl, \tag{27}$$

where $\mathsf{H}(V)$ is the entropy of $V$, $\mathsf{Pr}(v|\boldsymbol{l})$ can be computed thanks to Bayes as $\frac{\mathsf{f}(\boldsymbol{l}|v)\,\mathsf{Pr}(v)}{\sum_{v^* \in \mathcal{V}} \mathsf{f}(\boldsymbol{l}|v^*)\,\mathsf{Pr}(v^*)}$ and we use the same shortcut notation as in Section 2.3.1. For example, in a divide-and-conquer side-channel analysis targeting the S-boxes of the first AES round, the intermediate variables are the S-box output bytes $Z_i$. Since the AES S-box is bijective, information on $Z_i$ directly translates into information on the corresponding key byte $K_i$ given the knowledge of the corresponding plaintext byte $X_i$, as reflected by the equality $\mathsf{MI}(K_i; X_i, \boldsymbol{L}) = \mathsf{MI}(Z_i; \boldsymbol{L})$.

We note that the use of such an information theoretic metric in a security context may look surprising at first sight since, given a plaintext/ciphertext pair, there is essentially no entropy left in an AES key (which can then be recovered by exhaustive search with good probability). The necessary clarification is that in the side-channel analysis context, information theoretic metrics in general and the mutual information in particular are aimed to capture the amount of leakage on intermediate computations considered in isolation. Quantifying how this local information collected for many intermediate variables can be exploited by computationally-bounded adversaries is then the goal of security metrics.

(*Exercise 1.7 is suggested in order to further illustrate the previous content.*)

It turns out we already encountered security metrics as well. For example, Figure 10 implicitly suggests to compare attacks based on the number of traces to recover a key. Yet, it remains to determine what does it mean to recover the key. A first natural option for this purpose is to define a success as a situation where the attack directly outputs the correct key. But an attack outputting a set of $\ell$ keys including the correct one, next denoted as $\mathcal{K}_\ell$, may be sufficient to compromise security as well. It leads to the following definition of level-$\ell$ success rate:

$$\mathit{SR}_\ell(q_\mathsf{a}) = \mathsf{Pr}\left(\mathsf{A}_{\mathsf{kr}}^{\mathsf{AES}_k(\cdot) \rightsquigarrow \bar{\boldsymbol{l}}} \rightarrow \mathcal{K}_\ell, k \in \mathcal{K}_\ell | k \leftarrow \$\right), \tag{28}$$

where $\mathsf{A}_{\mathsf{kr}}^{\mathsf{AES}_k(\cdot) \rightsquigarrow \bar{\boldsymbol{l}}}$ is a side-channel key recovery adversary having access to $q_\mathsf{a}$ leakage traces of an AES implementation.* It can be defined similarly with a decryption oracle. For simplicity, we for now stick with an adversary leveraging known and uniformly distributed plaintexts. In the context of divide-and-conquer side-channel attacks, the success rate can then be estimated for each key byte independently and multiplied in order to obtain a success rate for the full key.

* We do not use the order-$o$ success rate terminology of [SMY09] to avoid confusion with the security order notion needed to analyze countermeasures in the second part of the book.

Alternatively, one can quantify the security of a leaking implementation with the key rank introduced in Section 2.5. In order to deal with the probabilistic nature of the side-channel attacks' outcomes, it is then necessary to additionally estimate a statistic based on the rank computed for several (ideally independent) attacks. The simplest solution for this purpose is to consider the average key rank, which then corresponds to the guessing entropy [Mas94]. The most complete one is to estimate all the quantiles of the key rank, which corresponds to the security graphs proposed in [VGS13]. We next mostly consider the median key rank, that we denote as $MR(q_{\mathsf{a}})$, which provides a good tradeoff between the simplicity of its estimation and the usefulness of the conclusions it leads to.

### 3.1.2 The SPA vs. DPA terminology

The main parameter influencing the previous security metrics is the attack data complexity $q_{\mathsf{a}}$. Yet, it may be interesting to refine this parameter by considering both the number of different plaintexts for which an adversary can observe the leakage, next denoted as $q_{\mathsf{a}}^{x}$, and the number of times the leakage of each plaintext is observed, next denoted as $q_{\mathsf{a}}^{\mathsf{r}}$. Assuming for simplicity that every plaintext is measured the same number of times, we have $q_{\mathsf{a}} = q_{\mathsf{a}}^{x} \cdot q_{\mathsf{a}}^{\mathsf{r}}$. This clarification directly leads to the definition of important classes of attacks that we next denote as Simple Power Analysis (SPA), SPA with repetition (repSPA) and DPA:

**Definition 4** (SPA, repSPA and DPA)**.** *A Simple Power Analysis is a side-channel attack where the adversary exploits the leakage of either KeS variables or KIS variables that can only be observed for a small number of different plaintexts (i.e., $q_{\mathsf{a}}^{x}$ is bounded), without repetition (i.e., $q_{\mathsf{a}}^{\mathsf{r}} = 1$). A Simple Power Analysis with repetition is the same attack where the KeS and KIS variables can be repeatedly observed (i.e., $q_{\mathsf{a}}^{\mathsf{r}}$ is chosen by the adversary). A Differential Power Analysis is a side-channel attack where the adversary exploits the leakage of KIS variables that can be observed for a chosen number of (possibly different) plaintexts $q_{\mathsf{a}}^{x}$.*

The intermediate computations of the AES rounds are a typical example of KIS variables. The intermediate computations of the AES key scheduling algorithm are a typical example of KeS variables. Figure 17 illustrates the difference between SPA, repSPA and DPA with the leakage caused by the S-box computation of two plaintext bytes $x_0^1$ and $x_0^2$, exploited in order to recover a key byte $k_0$. The size of the sets qualitatively represents the reduction of the number of possible values for the intermediate variable $z_0$ (i.e., the S-box output) and key byte $k_0$. The first

plaintext can only be measured once, leading to the light grey SPA leakage in the left part of the figure, where $q_{\mathsf{a}}^{x} = 1$ and $q_{\mathsf{a}}^{\mathsf{r}}(x_0^1) = 1$. The second plaintext can be measured multiple times, which allows reducing the noise in the measurements. It leads to the mid grey repSPA leakage in the left part of the figure, where $q_{\mathsf{a}}^{x} = 1$ and $q_{\mathsf{a}}^{\mathsf{r}}(x_0^2)$ is large. For each plaintext byte, these pieces of information on the intermediate variable $Z_0$ can then be translated into information on the target key byte $K_0$, by inverting the S-box and XORing with the known plaintext byte for all the candidate $K_0$ values, as reflected by the dashed arrows. It leads to the dark grey DPA leakage in the right part of the figure, where $q_{\mathsf{a}}^{x} = 2$. The figure therefore suggests an informal gradation of the types of attacks based on their strength (i.e., SPA is the least powerful, followed by repSPA and DPA).

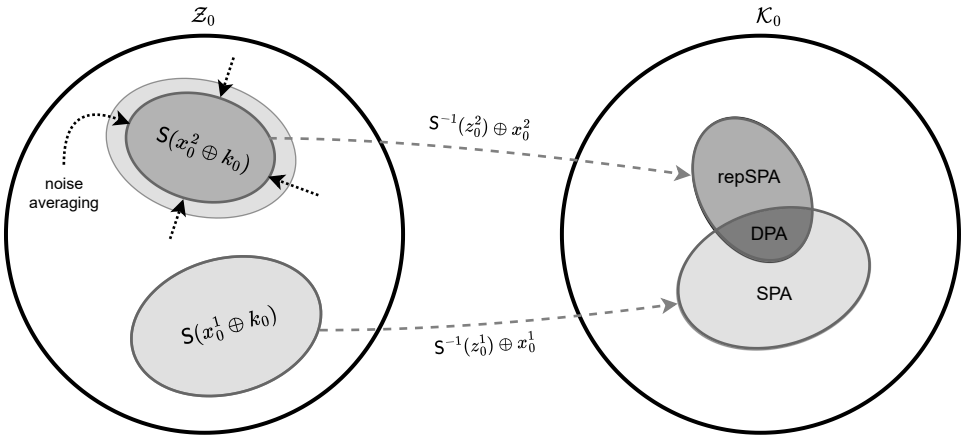



Figure 17: SPA, repSPA and DPA against a single S-box.

This explains our main focus on KIS variables, which are the most informative targets in a leaking AES implementation. We note also that we used the attack data complexity as main parameter for the metrics introduced in this section and the SPA vs. repSPA vs. DPA separation only depends on this parameter. Yet, the attack time complexity can also affect the security of a leaking implementation, in particular for the post-processing (key enumeration and ranking) part.

## 3.2 Bounds and shortcuts

We now consider the link between information theoretic and security metrics and show that thanks to information theoretic metrics, it is possible to bound the number of traces needed to recover a key with good probability, providing a shortcut towards faster security evaluations. This can be done formally for divide-and-conquer side-channel analysis and requires heuristics for analytical attacks.

### 3.2.1 IT bounds for divide-and-conquer attacks

For the unprotected implementations we consider in this first part of the book, it can be proved that the various information theoretic metrics introduced so far all provide similar hints on the complexity of the attacks using the corresponding side-channel distinguishers. The first option considered for this purpose was to use the correlation coefficient. For a divide-and-conquer attack targeting an AES S-box output byte $Z_i$, it was established in [Man04, SPRQ06] that the number of traces needed to recover the corresponding key byte $K_i$ with success rate *SR* using a univariate leakage variable $L_j$ can be approximated as:

$$Q_{\mathsf{a}}^{SR} \approx \frac{cst(SR, \mathcal{K})}{\rho(M_j^{z_i}, L_j)^2}, \tag{29}$$

where *cst* is a small constant depending on the target size and required success rate. Using Equation 6, it directly leads to a similar relation for the SNR:

$$Q_{\mathsf{a}}^{SR} \approx cst(SR, \mathcal{K}) \cdot \left(1 + \frac{1}{\mathsf{SNR}(j)}\right). \tag{30}$$

It was then highlighted in [MOS11] that in the context of standard DPA (i.e., DPA exploiting a divide-and-conquer approach with uniformly distributed plaintexts) and for univariate Gaussian leakage samples and models, we have:

$$\mathsf{MI}(Z_i; L_j) \approx -\frac{1}{2}\log_2\left(1 - \rho(M_j^{z_i}, L_j)^2\right) \approx -\frac{1}{2}\log_2\left(1 - \frac{1}{1 + \frac{1}{\mathsf{SNR}(j)}}\right), \tag{31}$$

which can be simplified into the standard MI bound from [CT06]:

$$\mathsf{MI}(Z_i; L_j) \leqslant \frac{1}{2}\log_2\left(1 + \mathsf{SNR}(j)\right). \tag{32}$$

The multivariate case has been treated by Cherisey et al. [dCGRP19]:

$$Q_{\mathsf{a}}^{SR} \leqslant \frac{\mathsf{f}(SR, \mathcal{K})}{\mathsf{MI}(K_i; X_i, \boldsymbol{L})}, \tag{33}$$

where it was also clarified that the function $\mathsf{f}(SR, \mathcal{K})$ is worth:

$$\mathsf{H}(K) - (1 - SR) \cdot \log_2 \Big( |\mathcal{K}| - 1 \Big) - \mathsf{H}(SR), \tag{34}$$

with $\mathsf{H}(SR) = -SR \cdot \log_2(SR) - (1-SR) \cdot \log_2(1-SR)$, due to Fano's inequality. Béguinot et al. finally extended this bound to level-$\ell$ success rates [BRM+26]:

$$Q_{\mathsf{a}}^{SR,\ell} \leqslant \frac{\mathsf{f}(SR, \mathcal{K}, \ell)}{\mathsf{MI}(K_i; X_i, \boldsymbol{L})}, \tag{35}$$

with the generalized $\mathsf{f}(SR, \mathcal{K}, \ell) = SR \cdot \log_2 \frac{SR}{\ell/|\mathcal{K}|} + (1 - SR) \cdot \log_2 \frac{1-SR}{1-\ell/|\mathcal{K}|}$.

Those results suggest that besides their interest for quantifying the leakage of an implementation, information theoretic metrics are relevant shortcuts to evaluate security against DPA, where the evaluator does not know in advance the number of measurements needed to recover the key with confidence.

However, it is also important to note that the formulas in this section are all for asymptotic values (i.e., perfect estimations) of the correlation, SNR and MI metrics, motivating our investigation of estimation issues in Section 3.3.

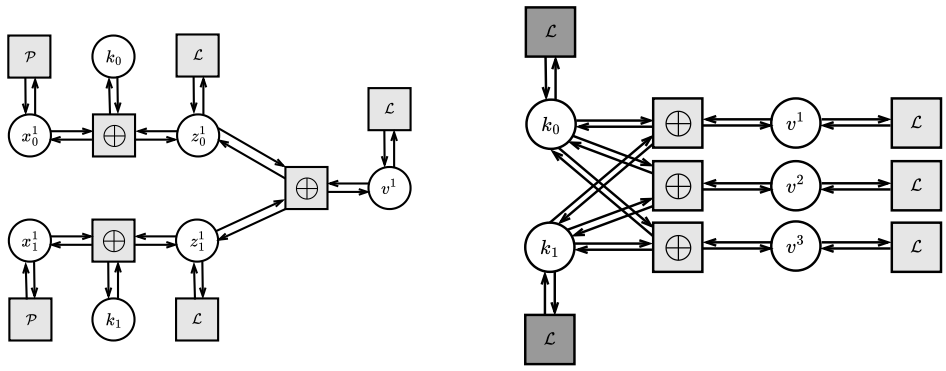
(*Exercises 1.8, 1.9 & 1.10 are suggested to further illustrate the previous content.*)

### 3.2.2 IT bounds for analytical attacks

Obtaining similar bounds for the (heuristic) SASCA described in Section 2.4.1 is unsuprisingly more difficult. Yet, reasonably simple intuitions can be obtained by modeling these attacks from a coding theory viewpoint [GGSB20], using the so-called Local Random Probing Model (LRPM) that we outline next.

At high-level, the LRPM uses the same factor graph as a SASCA. But rather than propagating local information as list of probabilities, it summarizes this information as a single (normalized) mutual information value. This leads to much faster running times than decoding with the BP algorithm. Such a gain comes at the cost of being an evaluation tool. Similar to the difference between key enumeration and rank estimation, the LRPM only provides an approximation of the attack

complexity and requires a worst-case evaluation setting with strong profiling capabilities. The name LRPM comes from its proximity with the Random Probing Model (RPM) used in masking security proofs [DDF19], restricted to local information propagation rules. It can also be viewed as an erasure channel [CT06], where we heuristically interpret the normalized MI as an erasure probability.




(a) Single-plaintext factor graph. (b) Three-plaintext factor graph.

Figure 18: Factor graph of Figure 11 adapted to the LRPM.

We illustrate the LRPM with the same example as for SASCA in Section 2.4.1. For this purpose, we first simplify the factor graph of Figure 11 into the factor graph of Figure 18a. At initialization, the public factor nodes (denoted with $\mathcal{P}$) have a normalized mutual information of 1, carried by the messages $\underset{\mathcal{P}\to z_i}{\varepsilon}$, the side-channel factor nodes connected to $z_0^1$ and $z_1^1$ have a normalized mutual information of $\lambda_Z = \frac{\mathsf{MI}(Z_i;\boldsymbol{L})}{8}$, carried by messages $\underset{\mathcal{L}\to z_i}{\varepsilon}$, and the side-channel factor node connected to $v^1$ has a normalized mutual information of $\lambda_V = \frac{\mathsf{MI}(V^1;\boldsymbol{L})}{8}$, carried by a message $\underset{\mathcal{L}\to v^1}{\varepsilon}$. In all cases, the denominator corresponds to the normalization by $\mathsf{H}(Z) = \mathsf{H}(V) = 8$, which is the size of the AES S-box.

Such a factor graph can then be directly "decoded" by adapting the message update rules of the BP algorithm in order to combine the normalized MI values. For the udating the variable-to-factor messages, the Hadamard product of probability vectors is turned into a sum of normalized MI values:

$$\underset{v\to f}{\varepsilon} = \sum_{f' \in \mathsf{neighbors(v)} \setminus f} \underset{f'\to v}{\varepsilon}. \tag{36}$$

For updating the factor-to-variable messages, we bound the normalized information on the sum of two (or more) random variables by the following product:

$$\underset{f \to v}{\varepsilon} = \prod_{v' \in \text{neighbors(f)} \setminus v} \underset{v' \to f}{\varepsilon}. \tag{37}$$

For both update rules, we additionally upper bound the normalized MI to one.

The example is generalized to the DPA setting by connecting the factor graphs corresponding to $q_{\mathsf{a}} = 3$ plaintexts in Figure 18b. In this case, we can further optimize the modeling by observing that for the variables that are divide-and-conquer targets (i.e., $z_0$ and $z_1$ in our example), the leakage of $q_{\mathsf{a}}$ plaintexts can be directly set to $q_{\mathsf{a}} \cdot \lambda_Z$ thanks to the (additive) bound of the previous section. Such an easy-to-exploit leakage is tagged by dark grey boxes in the figure.

*Cautionary note.* The LRPM is heuristic and can be optimistic for the adversary. For example, a single-bit leakage on two operands propagates through the XOR in the LRPM while it may not in a SASCA (if it has two different positions). It also ignores the possibly difficult convergence of SASCA when applied to factor graphs with cycles or loops. So the concrete security levels it provides must be interpreted with care. By contrast, the LRPM generally provides good hints on the factor of improvement that analytical attacks provide over divide-and-conquer ones (also predicated with IT bounds), which can then be used as a security margin.

(*Exercise 1.11 is suggested in order to further illustrate the previous content.*)

## 3.3 Computing the metrics

So far we ignored the concrete question of how to estimate information theoretic and security metrics for possibly large measurements' datasets. In this section, we tackle the theoretical and practical challenges raised by this question.

### 3.3.1 Evaluating the mutual information

There are two main difficulties to evaluate the mutual information. On the one hand, computing it based on Equation 27 is impossible since the distribution of the leakage vectors is unknown. On the other hand, its direct estimation is notoriously difficult, especially for long leakage traces with many samples [Pan03].

In this section, we therefore follow an alternative path, which is to estimate the amount of information that can be extracted from the models introduced in

Section 2.3. We denote this quantity as the Perceived Information (PI) [RSV+11], and the approach to estimate it as indirect. Compared to the direct approach, it has the advantage of bringing useful feedback on the profiling phase of a side-channel analysis [SKS09], of leading to information estimates that can be directly exploited by concrete attacks and therefore of scaling with the size of the traces as well as these attacks. Its main drawback is a possible deviation from worst-case security evaluations, since profiled models can be imperfect due to estimation errors (i.e., a lack of leakage traces for profiling) or assumption errors (i.e., an incorrect choice of modeling tools), which we discuss in Section 3.4.

The PI is defined similarly as the MI with as main difference that the log probabilities that are averaged are not coming from the true (unknown) leakage distribution but from a model estimated with a set of profiling traces $\mathcal{L}_\mathsf{p}$:

$$\mathsf{PI}(V;\boldsymbol{L}) = \mathsf{H}(V) + \sum_{v\in\mathcal{V}} \mathsf{Pr}(v) \cdot \int_{\boldsymbol{l}\in\mathcal{L}} \mathsf{f}(\boldsymbol{l}|v) \cdot \log_2 \underbrace{\hat{\mathsf{Pr}}(v|\boldsymbol{l})}_{\text{estimated model}} \; dl, \tag{38}$$

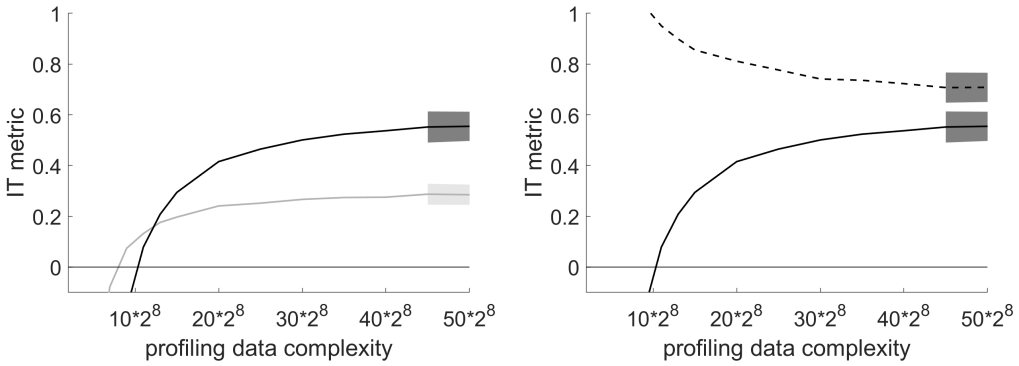
where the hat notation is used to reflect the estimation of the model from which the probabilities are inferred, which may be imperfect (i.e., affected by estimation or assumption errors). This quantity remains difficult to compute since it involves the true leakage distribution $\mathsf{f}(\boldsymbol{l}|v)$ at the left of the logarithm. Yet, this time it is easy to estimate by replacing this true leakage distribution with an estimation of the empirical distribution that converges towards the true distribution:

$$\tilde{\mathsf{PI}}(V;\boldsymbol{L}) = \mathsf{H}(V) + \sum_{v\in\mathcal{V}} \mathsf{Pr}(v) \cdot \underbrace{\sum_{\boldsymbol{l}\in\mathcal{L}_\mathsf{t}(v)} \frac{1}{|\mathcal{L}_\mathsf{t}(v)|}}_{\text{estimated empirical}} \cdot \log_2 \hat{\mathsf{Pr}}(v|\boldsymbol{l}), \tag{39}$$

where the tilde notation is used to reflect the estimation of the PI metric based on a set of test samples $\mathcal{L}_\mathsf{t}$ (which must be different from the profiling set $\mathcal{L}_\mathsf{p}$). It is shown in [BHM+19] that the PI is a lower bound of the mutual information and it converges towards the true mutual information if the model is perfect.

Learning curves corresponding to the univariate and bivariate TA models used in Section 2.3.4 (Figures 10d and 10e) are given in Figure 19a. They show the evolution of the estimated PI in function of the profiling data complexity $q_\mathsf{p} = |\mathcal{L}_\mathsf{p}|$, where we use the same number of traces for all templates. For the size of the test set $\mathcal{L}_\mathsf{t}$, a reasonable rule-of-thumb is to take it equal to $|\mathcal{L}_\mathsf{p}|$. For each attack, the learning curves are averaged over 100 profiling phases.

We can observe that for too small profiling sets, the PI of both models is negative (indicating models that are not sufficiently accurate to lead to successful key recoveries). Furthermore, we see that the PI of the univariate TA converges faster than the bivariate one, towards an asymptotic value that is roughly halved. The point where both curves interesects reveals the amount of profiling needed for the bivariate TA to become more informative than the univariate one.




(a) Univariate PI vs. bivariate PI. (b) Bivariate PI vs. bivariate TI.

Figure 19: Estimation of information theoretic metrics.

Such curves can be used in order to gauge the tradeoff between the profiling data complexity (quantified on the X axis) and the attack data complexity (which is inversely proportional to the PI of the models reported on the Y axis). Given the variance of the models, it is advisable to add a confidence interval for the plots' rightmost points. Relying on a Gaussian assumption or using bootstrapping can be used for this purpose [ET93]. For example, the shaded (grey) areas of Figure 19a correspond to the interquartile range of our PI estimates.

*Efficient use of the measurements.* Collecting leakage traces is a time-consuming process and it is beneficial for side-channel evaluators to make the best use of a small profiling set. For this purpose, it is also advisable to leverage $k$-fold cross-validation. That is, first collect a single set $\mathcal{L}_{\mathsf{pt}}$ for profiling and test and split it in $k$ equal parts $\mathcal{L}_{\mathsf{pt}}^{(\mathrm{i})}$. Next, define $k$ overlapping profiling sets $\mathcal{L}_{\mathsf{p}}^{(\mathrm{j})} = \underset{\mathrm{i}\neq\mathrm{j}}{\cup} \mathcal{L}_{\mathsf{pt}}^{(\mathrm{i})}$ and the corresponding test sets $\mathcal{L}_{\mathsf{t}}^{(\mathrm{j})} = \mathcal{L}_{\mathsf{pt}}^{(\mathrm{j})}$. The sample metric can then be repeatedly computed $k$ times by estimating a model with $\mathcal{L}_{\mathsf{p}}^{(\mathrm{j})}$ and testing it with $\mathcal{L}_{\mathsf{t}}^{(\mathrm{j})}$. Cross–validation protects evaluators from obtaining too optimistic sample metric values due to overfitting, since the test computations are always performed with an in-

dependent data set. Finally, the $k$ outputs can be averaged to get an unbiased metric estimate, and their spread characterizes the results' accuracy.*

*Bounding the learnable information.* Another relevant quantity for security evaluators is the "learnable information" that is the amount of information that remains to be learned by increasing the profiling data complexity of a model (i.e., the supremum of the PI). It is shown in [MCHS23] that on average over the profiling phases, and under some conditions on the modeling tool, the following Training Information (TI) is an upper bound of the learnable information:

$$\tilde{\mathsf{TI}}(V;\boldsymbol{L}) = \mathsf{H}(V) + \sum_{v\in\mathcal{V}} \mathsf{Pr}(v) \cdot \underbrace{\sum_{\boldsymbol{l}\in\mathcal{L}_{\mathsf{p}}(v)} \frac{1}{|\mathcal{L}_{\mathsf{p}}(v)|}}_{\text{overfitted empirical}} \cdot \log_2 \hat{\mathsf{Pr}}(v|\boldsymbol{l})\, dl, \tag{40}$$

where the overfitted empirical is just the empirical distribution estimated with the profiling set (rather than with the test set as in Equation 39). The TI and the PI of the bivariate TA of our simulated experiments are given in Figure 19b.

*Variability as an additional security margin.* We finally note that, in theory, the estimations of this section can be performed with profiling and test sets coming from the same devices or from different devices. As discussed in [BS21], the first option is preferable in a worst-case evaluation setting (and may provide a small security margin in case the leakage functions of different devices differ).

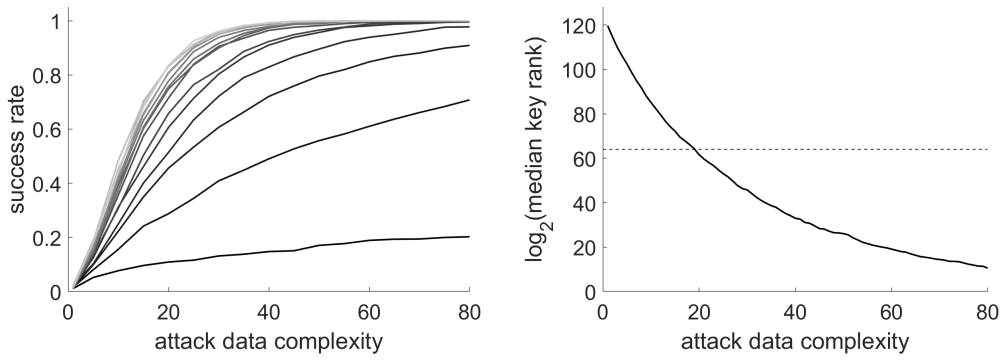
(*Exercise 1.12 is suggested in order to further illustrate the previous content.*)

### 3.3.2 Evaluating security metrics

The evaluation of security metrics is conceptually straightforward but time- and data-intensive. It requires to launch several experiments in order to sample the metrics from the attack set $\mathcal{L}_{\mathsf{a}}$ that has to be independent of (and different from) the previous training and test sets. While the training and test sets are generated from random plaintexts and keys (as for the estimation of the SNR in Section 2.2.1), the attack set contains random plaintexts for a few fixed keys to be recovered by side-channel analysis. We illustrate the estimation of security metrics in Figure 20. In the left part, we show sixteen success rate curves corresponding to the 16 AES S-boxes targeted with a bivariate TA. Each of them is estimated from 100 independent experiments. They differ because we used a different SNR for

* The same strategy can be applied to the profiled correlation coefficient of Section 2.2.2.

every S-box (to make them distinguishable on the figure). In the right part, we show an estimation of the median key rank from the same 100 experiments.




(a) Success rate per S-box. (b) Full key median rank.

Figure 20: Estimation of security metrics.

We note that re-sampling techniques could theoretically be used to reduce the amount of data needed to estimate security metrics. Yet, applying these techniques is made more difficult by the requirement to target fixed keys, since nothing theoretically precludes that measurements show key-dependencies.* As a result, we recommend the estimation of the median key rank from at least a few independent attacks (i.e., Figure 20b) as a generally good evaluation practice.

We wrap up this section by discussing the complementarity of information theoretic and security metrics in the DPA setting with Figure 21. It illustrates how the success rate of an attack against a single S-box scales in function of both the profiling data complexity and the attack data complexity. The framework of Figure 16 allows an evaluator to "factorize" this plot in two parts. First, the profiling data complexity is analyzed thanks to the information theoretic analysis of Figure 19. It has the advantage of enabling easier comparisons between different profiling methods since finding where one becomes preferable to another is easier with 2D curves than 3D plots. It is also easier to estimate, especially as the attack data complexity increases. Second, the security of the implementation can be evaluated for the best model using the minimum profiling data complexity as in Figure 20b, or bounded thanks to the tools of Section 3.2. By contrast, in the case of SPA and repSPA (i.e., when the number of attack traces is fixed), the

* We nevertheless suggest in Exercise 1.2 that this risk is limited.

interest of information theoretic metrics vanishes and it becomes preferable to directly evaluate the profiling and attack data complexities by estimating security metrics for the (small) number of plaintexts $q_a^x$ that are used in the attack.

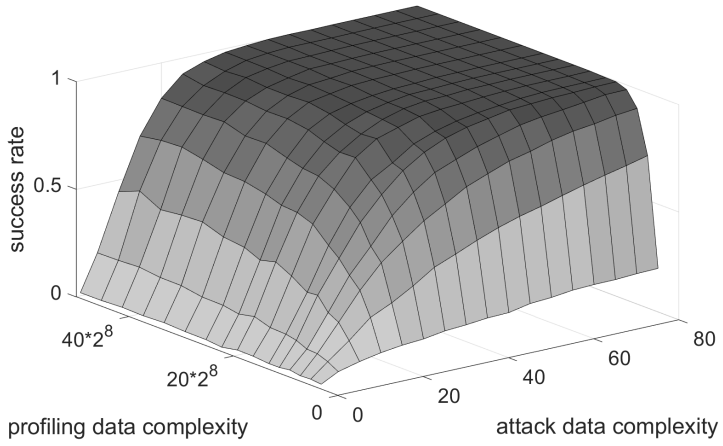



Figure 21: Direct evaluation of the profiling and attack data complexities.

### 3.3.3 Shortcut limitations

Despite their convenience, it is important to keep in mind that the shortcut formulas of Section 3.2 are not aimed to replace the concrete estimation of security metrics. They should rather be seen as an interesting tool for design space exploration and as a proxy to analyze the scaling trends of a leaking implementation, for various reasons. First, the information theoretic metrics they leverage are average metrics, which can therefore hide the fact that a model for some intermediate values has been poorly estimated. Second, the (linear) bound that is assumed for combining the leakage of multiple plaintexts is general and can be more or less optimistic, depending on the target operations, leakage functions and level of success rate targeted. It is for example well-known that for exactly the same leakage model, the level-1 success rate of an attack targeting the AES S-box input and output differ [Pro05]. More generally, it is unavoidably hard to summarize the security of an implementation with a single tool or metric. Comprehensive security evaluations therefore highly benefit from assessing all the steps of the workflow in Figure 4, and to check that they lead to consistent conclusions.

## 3.4 Discussion and related works

We finally step back to the tools and metrics introduced in this part of the book. In this respect, a first question relates to the necessity of profiling for worst-case evaluations, since most of the tools in Chapter 2 are profiled. For this purpose, we refer to Whitnall et al. who investigated whether generic DPA, defined as non-profiled attacks that do not require any a-priori knowledge about a devices' leakage characteristics, can be successful. They showed the following:

**Proposition 1** (Impossibility of generic DPA [WOS14])**.** *There is no generic DPA able to distinguish the correct key from an alternative hypothesis in a side-channel analysis if the composition of the function connecting the key to the target intermediate variable of the attack and the leakage function is injective.*

*Proof sketch.* This proposition formalizes the next two main observations.

First, generic DPA must be based on the least demanding type of modeling, which approximates the leakage function up to nominality [Ste46]. Therefore, it requires a statistic that interprets the samples as arbitrary labels only, which corresponds to the partition-based distinguishers introduced in [SGV08].

These generic-compatible distinguishers exploit side-channel leakage by organizing the measured samples according to a key-dependent model of some target intermediate value. An illustration is given in Figure 22 for 16 exemplary leakage samples obtained for the output of a 4-bit S-box $z = \mathsf{S}_4(x \oplus k)$, assuming a (noise-free) Hamming weight leakage model that perfectly corresponds to the actual leakage function, for the correct key guess and an incorrect key guess.

The partitions are then evaluated thanks to a statistical tool that can analyze nominal data, such as MIA [GBTP08] or variations thereoff [PR09, VS09, WOM11]. The attack succeeds when the correct key guess leads to the best partition (i.e., informally, the partition that groups leakage samples that are most similar).

Second, and contrary to the example above, generic DPA must leverage an identity leakage model in order to capture the absence of a priori physical assumptions. However, when such a generic leakage model is used, the partitions for the correct and incorrect key guesses are both made of columns that have perfect (intra-column) similarity. Figure 23 illustrates such partitions for 32 exemplary leakages obtained for the output of $S_4$, assuming that the generic (identity) leakage model perfectly corresponds to the actual leakage function.

As a result, combining a generic-compatible distinguisher with a generic leakage model cannot lead to successful side-channel attacks, even if this generic

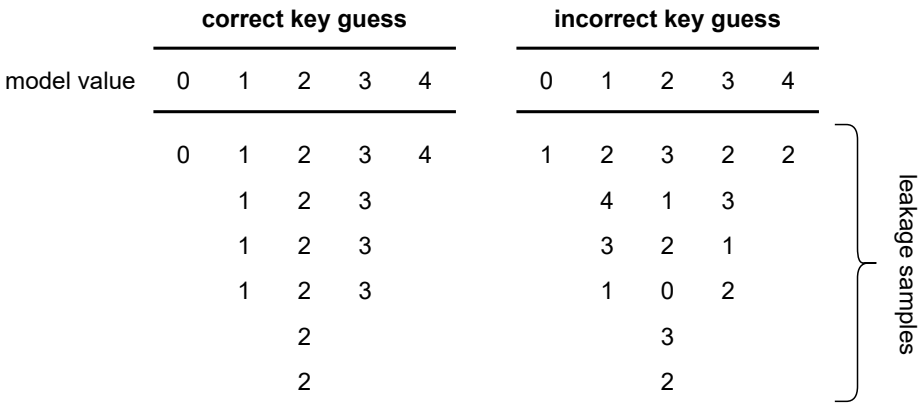



Figure 22: Partition-based distinguisher with non-injective leakage model.

leakage model is perfect. Distinguishing the correct key guess from an incorrect one requires either (non-injectivity) assumptions on the target operation or leakage function, or to analyze the leakage samples under a different scale of measurements (e.g., proportional as in CPA or direct as in profiled TAs). □

For a similar reason, the non-profiled extensions of LR-based attacks cannot succeed with a full (generic) basis either [DPRS11, WOS14, WYS+18].

We note that Proposition 1 only applies to the so-called standard DPA attacks formalized in [MOS11] and not to collision attacks such as [SLFP04, LMV04, Bog07a, MME10, GS13, GG19]. However, these attacks are not generic either in the sense that they can only succeed under some assumption regarding the leakage of different operations, so that collisions can indeed be detected.

Such a negative result shows that non-profiled side-channel security evaluations inherently and unavoidably come with a risk of security overstatements. But it also raises the complementary question whether there could be a practical gap between profiled and non-profiled attacks, implying that profiled evaluations could be too conservative. In other words, can the results of profiled attacks generally be matched by non-profiled ones given more time and efforts, or are there relevant examples of profiled evaluations that cannot be reproduced without knowledge of the implementation and strong profiling capabilities? We informally answer to this question with Table 3 which summarizes the optimality of the evaluation steps of Section 2 in the worst case and uninformed (non-profiled) evaluation settings. While some of the steps (e.g., modeling and extraction) be-

**correct key guess**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

**incorrect key guess**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 5 | 12 | 8 | 15 | 1 | 14 | 0 | 10 | 6 | 3 | 13 | 2 | 11 | 9 | 4 |
| 7 | 5 | 12 | 8 | 15 | 1 | 14 | 0 | 10 | 6 | 3 | 13 | 2 | 11 | 9 | 4 |

Figure 23: Partition-based distinguisher with injective leakage model.

come marginally more difficult, it suggests that for the unprotected implementations we consider in this part of the book, none of the steps (which mostly involve first-order statistics) is expected to lead to significantly harder problems.

This table informally answers another question regarding the indirect approach chosen to estimate the mutual information in Section 3.3.1. Namely, since the PI is only a lower bound of the MI, could there be a large gap between them? In other words, and despite working in a profiled setting, could it be that the models estimated with the tools of Chapter 2 deviate from the true leakage distribution, leading to a PI significantly lower than the MI (i.e., security understatements). We posit that in the unprotected implementation context, this risk is again quite limited due the relatively simple nature of the leakage distributions to analyze.

In general, we therefore presume that side-channel security evaluations performed under strong adversarial conditions, as advocated in this book, can be used as a timesaver for unprotected implementations, with limited chance that they drive overly optimistic nor conservative conclusions. This heuristic assumption is nailed down by the following conjectures, illustrated in Figure 24:

**Conjecture 1** (Worst-case attack conjecture)**.** *Profiled side-channel evaluations performed in worst-case adversarial conditions can provide a tight estimate/bound of the security level against the (unknown) optimal side-channel adversary.*

| evaluation step | | evaluation setting: *worst-case* | | evaluation setting: *uninformed (unprotected)* |
|---|---|---|---|---|
| measurement & pre-processing | | ● | < | ● |
| detection & mapping | | ○ | | ○ |
| modeling & extraction | | ◔ | | ◔ |
| information exploitation | D&C | ○ | | ○ |
| | analytical | ● | < | ● |
| post-processing | | ○ | | ◔ |

Table 3: Informal evaluation of the attack steps' optimality: ○ ≈ close to optimal attack step; ◔ ≈ may become computationally hard to be optimal; ● ≈ involves hard to assess heuristics; ● ≈ computationally hard and heuristic step.

**Conjecture 2** (Uninformed profiling conjecture)**.** *The best side-channel attack data and time complexities (i.e., the security level) indicated by a profiled evaluation performed under worst-case adversarial conditions can be reached in an uninformed evaluation context with an amount of evaluation data and time that is not significantly larger than required in the worst-case evaluation setting.*

We note that the first conjecture is needed to connect concrete side-channel security evaluations with the provable security guarantees of the countermeasures investigated in Part II of the book. As for the second conjecture, it states that the problem of profiling (possibly on-the-fly) with limited implementation knowledge is not significantly harder (in data and time) than the problem of attacking this implementation once a well-profiled model is given to the adversary. The word significant here has to be understood in the cryptographic sense: what we mean is that the profiling problem is not hard in some security parameter.

Both conjectures naturally become more questionable in the context of protected implementations and will be re-discussed in Part III of the book.

We close this part with the following definition of side-channel attack:

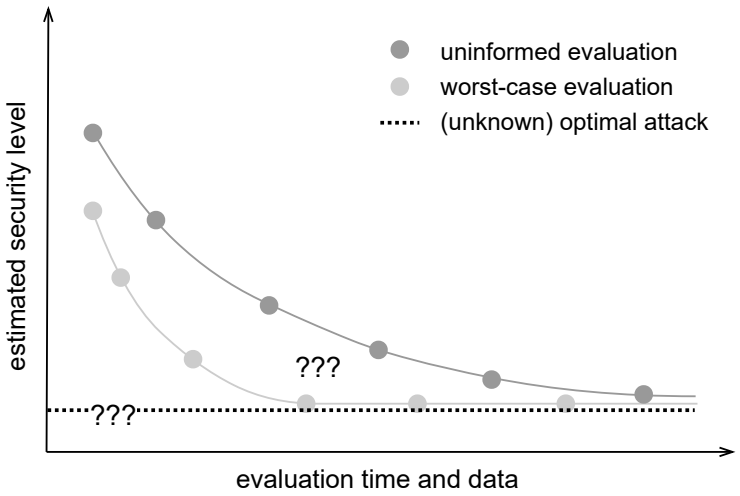



Figure 24: Evaluation conjectures for unprotected implementations.

**Definition 5** (Side-channel attack)**.** *A side-channel attack is a cryptanalysis technique that exploits the information leaked by the physical measurements of a cryptographic implementation in order to break a given security notion.*

The two key words of this definition are *information* and *exploit*: they suggest two important approaches for designing countermeasures against side-channel attacks that we will investigate in the next parts of the book: reducing the information leakage and making its exploitation (computationally) difficult.

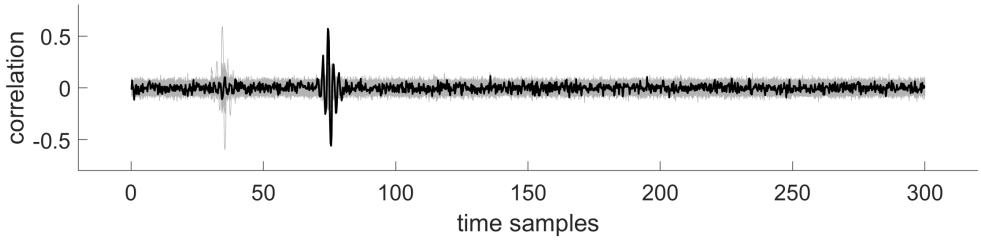
**Related works.** The separation between information theoretic and security metrics is revisited through the prism of machine learning in [PHJ+19]. The authors make the same observation as [SPAQ06] that security metrics (like the accuracy) can be deceptive in the DPA setting where the number of traces needed to perform a key recovery is not known in advance. Finer-grain bounds for the data complexity of CPA (relying on a finer-grain characterization of the S-box) are described in [FLD12]. Rivain analyzes the success rate of the Gaussian side-channel in [Riv08]. Message propagation rules for multiplications in the LRPM are given in [CS19]. Cristiani et al. and Chowdury et al. investigate direct approaches for estimating the mutual information [CLM20, CRBO24]. The optimality of the side-channel distinguishers introduced in Chapter 2 and used in order to estimate the metrics of Chapter 3 is discussed in [HRG14, BGH+15], backing up the methodology proposed in this part of book when applied to unprotected implementations. Finally, a

detailed investigation of generic-compatible distinguishers relying on linear regression, together with the conditions under which they succeed, can be found in [Bou26].

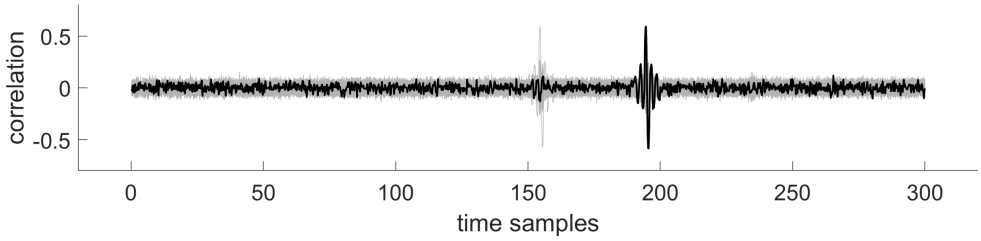
# Exercises

## Paper-and-pencil

**Exercise 1.1** (*Ghost peaks*). Figure 25 illustrates the results of a profiled CPA similar to the one of Section 2.2.2 but targeting the S-boxes' inputs. Explain the "ghost peaks" (i.e., the non-negligible correlation that appears for incorrect key candidates) observed a few cycles before the manipulation of the S-boxes' inputs.




(a) First S-box.




(b) Second S-box.

Figure 25: Correlation between a profiled model for the AES S-boxes' inputs ($\forall$ key byte candidates) and leakage traces for the correct key byte ($q = 1024$).

**Exercise 1.2** (*Weak keys*). Consider a CPA targeting a KIS variable (e.g., the first-round AES S-boxes) with uniformly distributed plaintexts. Does it have weak keys (i.e., keys that are significantly easier to recover via CPA)? Does the answer change with a SPA targeting a KeS variable (e.g., a byte of the AES master key)?

**Exercise 1.3** (*Fixed vs. fixed detection*). The TVLA methodology of Section 2.2.3 works by comparing the mean of a leakage set with fixed plaintext and key with the one of a leakage set with random plaintexts and keys. Assume an hypothetical implementation leaking the Hamming weight of the first-round AES S-boxes. On average, would a variant of this detection test comparing two leakage sets with fixed plaintext and key be more or less efficient in terms of data complexity?

**Exercise 1.4** (*CPA vs. Gaussian TA*). Consider the profiled CPA of Section 2.2.2 and the Gaussian TA of Section 2.3.1 restricted to a univariate model. Assume the homoscedasticity assumption is fulfilled, plaintexts are uniformly distributed and both attacks use the same perfect model. Do these attacks have significantly different data complexities? What does it imply for the respective importance of using a good (profiled) model vs. choosing a good statistical distinguisher?

**Exercise 1.5** (*Non-informative samples*). Assume an evaluator uses the multivariate Gaussian TA of Section 2.3.1 in order to assess the security level of a leaking implementation and has unlimited profiling (data and time) complexity. Does the incorporation of non-informative samples in the model, in combination with POIs showing significant SNR, affect the attack data complexity? Would the answer change with a profiling phase that is limited in data and time?

**Exercise 1.6** (*Rank estimation*). Consider the rank estimation described in Algorithm 1 applied to sixteen AES key bytes. How does its (well estimated) result compare to the crude approximation where the sixteen ranks of the key bytes are multiplied (i.e., is this product of byte ranks conservative or risky)?

**Exercise 1.7** (*HW leakage*). Provide an upper bound for the mutual information between a uniformly random $\eta$-bit value and its Hamming weight.

*Hint.* A discrete random variable has maximum entropy if uniformly distributed.

**Exercise 1.8** (*Algorithmic noise*). Consider a side-channel adversary targeting the first S-box of increasingly parallel hardware implementations of the AES, as depicted in Figure 26. In the 8-bit case (Figure 26a), the adversary receives sixteen leakage samples $l_i^8 = \mathsf{HW}\big(\mathsf{S}(x_i \oplus k_i)\big) + r_i^8$, with $0 \leqslant i \leqslant 15$. In the 32-bit case

(Figure 26b), she receives four leakage samples of the form:

$$l_i^{32} = \left( \sum_{j=4i}^{4i+3} \mathsf{HW}\big(\mathsf{S}(x_j \oplus k_j)\big) \right) + r_i^{32},$$

with $0 \leqslant i \leqslant 3$. In the 128-bit case (Figure 26c), she receives a single sample:

$$l_0^{128} = \left( \sum_{j=0}^{15} \mathsf{HW}\big(\mathsf{S}(x_j \oplus k_j)\big) \right) + r_0^{128}.$$

Assume the adversary knows that the Hamming weight model is perfect so that we can write $L_i = M_i + R_i$ and the implementation has low physical noise so that $\rho(M_i, L_i) \approx 1$ for all architectures. Further assume that the model is split into an exploitable (signal) part that corresponds to the fraction of bits from the target intermediate computation that can be predicted with an 8-bit guess, which we denote as $M_i^{\mathsf{s}}$, and an unpredictable part that corresponds to the other bits manipulated in parallel, which we denote as $M_i^{\mathsf{n}}$ and is usually referred to as "algorithmic noise", so that $M_i = M_i^{\mathsf{s}} + M_i^{\mathsf{n}}$. Finally assume that the 8-bit architecture is broken with good success rate in 8 traces. How does the attack data complexity (to recover a single key byte) scale for the 32-bit and 128-bit architectures?

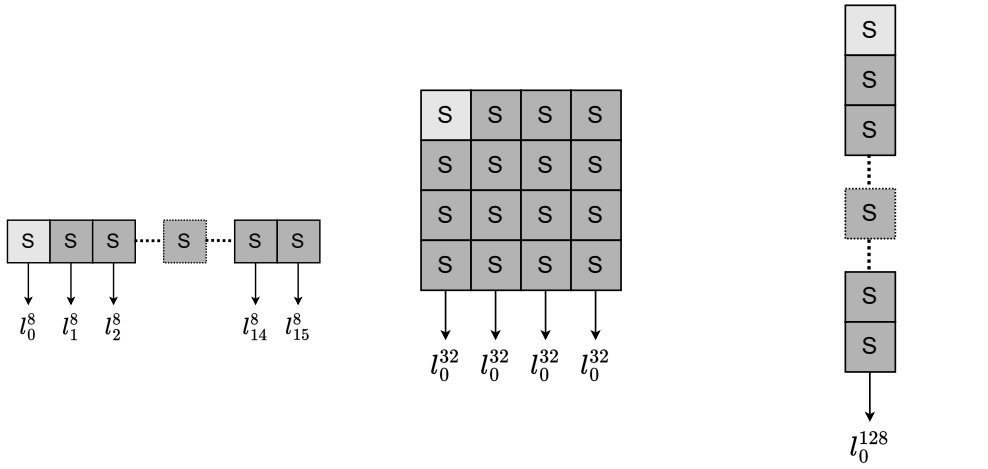
*Hint.* Use the correlation chain rule of Section 2.2.2 and the following lemma:

**Lemma 2** (Maximum correlation for sums of iid variables)**.** *The correlation coefficient between the sum of $\eta$ independent and identically distributed random variables and the sum of the first $m < \eta$ of these equals $\sqrt{m/\eta}$.*

What does the answer imply for using algorithmic noise as a security parameter?

How does the answer evolve if the physical noise in the implementation increases so that $\rho(M_i, L_i) < 1$? Consider the cases where (*i*) the physical noise variance $\mathsf{Var}(R_i)$ grows with the size of the architecture so that the SNR of the full $\eta$-bit computations (i.e., ignoring algorithmic noise) is constant for all architectures, and (*ii*) the physical noise variance is constant for all architectures.

**Exercise 1.9** (*Chosen plaintexts*). Consider a DPA against the 128-bit architecture of Figure 26c which, as discussed in Exercise 1.7, is affected by the algorithmic noise of fifteen S-boxes. Can a chosen-plaintext attack where these S-boxes' inputs are set to constant (to cancel the algorithmic noise) improve the attack data complexity, taking into account that the full (128-bit) key must be recovered.




(a) 8-bit architecture. (b) 32-bit architecture. (c) 128-bit architecture.

Figure 26: DPA against a single S-box with increasing algorithmic noise.

**Exercise 1.10** (*Large key guesses*). Consider a DPA against the 32-bit architecture of Figure 26b which, as discussed in Exercise 1.7, is affected by the algorithmic noise of three S-boxes. Can a 32-bit key guess improve the attack data complexity, taking into account that the full (128-bit) key must be recovered.

*Hint.* Assume that the results of Section 3.2.1 are tight and the function of Equation 34 obtained thanks to Fano can be used for the constant of Equation 29.

**Exercise 1.11** (*Analytical attacks*). Consider the information theoretic (LRPM-based) bounds given in Section 3.2.2 and the factor graph of Figure 18b. By which factor does the exploitation of the leakage obtained for the intermediate variables $Z_0$ and $V$ in a SASCA improve the data complexity of a divide-and-conquer attack exploiting the leakage obtained for the intermediate variable $Z_0$ only? For simplicity, assume that the normalized MI values $\lambda_Z$ and $\lambda_V$ are identical.

**Exercise 1.12** (*Mutual and Perceived Information*). Assume a side-channel security evaluator mistakenly enforces a pooled covariance matrix (as described in Section 2.3.1) despite the target implementation exhibits highly heteroscedastic (i.e., data-dependent) noise. What is the impact on the Mutual Information (MI), on the Perceived Information (PI) and on the attack data complexity?

## Hands-on

Programming problems based on actual side-channel measurements of an unprotected AES implementation, illustrating the content of this first part of the book, are available from: `https://github.com/simple-crypto/scale-one`. They are part of the SCALE-I training organized by the SIMPLE-Crypto association (see `https://www.simple-crypto.org/activities/scale/`).

# Part II

# Leakage-Resistance

# Introduction

The first part of the book showed how side-channel leakage can be used to extract (sensitive) information about a cryptographic implementation's intermediate operations and to turn this physical information into a key recovery attack against a block cipher. In this second part, we aim to introduce solutions to prevent such side-channel attacks and to ensure the basic functionalities of symmetric cryptography: authentication, encryption and authenticated encryption.

We follow a top-down approach for this purpose. We start with definitions of security with leakage, go on with authentication, encryption and authenticated encryption schemes that can better cope with leakage (i.e., mode-level countermeasures) and conclude with implementation-level countermeasures that must be used to secure these schemes against side-channel attacks. Mode-level and implementation-level countermeasures are complementary. The first ones aim to avoid that all the intermediate operations of a cryptographic primitive can be targeted with the strongest types of side-channel attacks (e.g., DPA). Such mode-level countermeasures come at the cost of updating cryptographic schemes. The second ones are generic solutions that can be applied to any algorithm.

The arguments we provide in this part of the book are of theoretical nature. Following the standard methodology of cryptographic research, we try to reduce the security of complete implementations to a few well-identified and falsifiable assumptions, and we use abstract models for this purpose. The computational model we consider for cryptographic implementations is essentially similar to the one we used for SASCA in Chapter 2.4.1 and their bounds in Section 3.2.2, where implementations are represented as a sequence of operations on intermediate (leaking) variables. The physical assumptions we rely on are of three types:

1. *Profiling.* For any variable $z$ manipulated by an implementation, we assume a perfect knowledge of the leakage function $\mathsf{L}(z) = \delta(z) + r$, with $\delta$ a deterministic (e.g., HW) function and $r$ an additive (e.g., Gaussian) noise.

2. *Uncertainty.* For any variable $Z$ manipulated by an implementation, we assume that $\mathsf{MI}(Z; L) < 1$, for example thanks to a sufficient noise.

3. *Independence.* For any variable $Z$ manipulated by an implementation, we assume that the deterministic part of the leakage function only takes $Z$ as argument and its noise is independent of the other leakage samples.

These assumptions are ordered from the least to the most critical. The profiling assumption is a primarily technical one. Since we aim to cover the strongest possible attacks, it is natural to give adversaries the knowledge that is needed to mount them. Beside, assuming a perfect knowledge of the leakage distribution is standard for existential results (e.g., reductions from security in one leakage model to security in another leakage model leveraging the simulation paradigm). Yet, we also conjecture that an imperfect knowledge of the leakage distribution is not harmful (even in practice) as long as the level of imperfection is similar for designers and adversaries. The uncertainty assumption is of more fundamental nature. Without a sufficient level of noise (or, in some cases, compression), nothing prevents an implementation secret to be leaked in full, making any security claim impossible. Concretely though, it is a relatively simple assumption to achieve, as already hinted in the previous part (e.g., Exercise 1.8). The next sections will formalize different models that capture this uncertainty assumption and can be used in security proofs. Their relations are discussed in Section 7. Finally, the independence assumption is both of fundamental nature and hard to achieve. This is because it is in general hard to ensure that physical phonemena taking place in close proximity (e.g., on the same circuit) are independent. How to ensure a sufficient level of independence will therefore be a core topic in the third part of the book, where we generally discuss how to ensure the physical assumptions needed for proofs to deliver their security guarantees in practice.

Given the more theoretical nature of the following chapters, the exercises we provide at the end of this part of the book are more balanced towards paper-and-pencil. However, a few programming problems are additionally provided in order to illustrate the challenge of ensuring our three physical assumptions.

# Chapter 4

# Security definitions

The previous chapters focused on side-channel key recovery attacks. It is however well-known that security against such attacks is not sufficient [KL14]. In this section, we therefore detail how to extend classical security definitions for symmetric cryptography, taking side-channel leakage into account. For this purpose, we start by completing our taxonomy of attacks to include distinguishing adversaries that are relevant to discuss impossibility results and confidentiality with leakage. We then define authentication and encryption with leakage, and conclude with the combination of these definitions for authenticated encryption.

## 4.1 Completing the attacks' taxonomy

A side-channel key recovery experiment matching the investigations in Part I of the book is described in Table 4, for a security parameter $n$. It is given for a Pseudo-Random Permutation (PRP) which can be instantiated with the AES, and includes a profiling oracle $\mathsf{Lprof}(.)$, which allows the adversary to collect traces for plaintexts, keys and ciphertexts of her choice.* The $\mathsf{L2}$ superscript indicates that the adversary has access to both encryption and decryption oracles with leakage. Using the shortcut formulas of Section 3.2, the (level-1) success rate of such side-channel key recovery attacks can be rewritten and approximated as:

$$\Pr\left(\mathsf{KR}^{\mathsf{L2}}_{\mathsf{PRP}}(n) \rightarrow 1\right) \approx \min\left(2^{-n+q_{\mathsf{a}}^{x} \cdot \Lambda(q_{\mathsf{a}}^{\mathsf{r}})}, 1\right), \tag{41}$$

with $q_{\mathsf{a}}^{x}$ the (adversarially-chosen) number of plaintexts for which leakage is observed and $\Lambda(q_{\mathsf{a}}^{\mathsf{r}})$ the (average) leakage obtained per plaintext, given that each

* As per the profiling assumption above, we assume a perfect profiling (i.e., $q_{\mathsf{p}} = \infty$).

| $\mathsf{KR}^{\mathsf{L2}}_{\mathsf{PRP}}$ experiment$(n)$ | |
|---|---|
| Initialization$(n)$<br>$k \leftarrow \$$ | Oracle $\mathsf{Ldec}(y)$<br>$\mathsf{PRP}_k^{-1}(y) \rightsquigarrow (x, \boldsymbol{l}_\mathrm{d})$<br>return $(x, \boldsymbol{l}_\mathrm{d})$ |
| Oracle $\mathsf{Lprof}(.)$ | Finalization<br>$\check{k} \leftarrow \mathcal{A}^{\mathsf{Lenc}(.),\mathsf{Ldec}(.)}_{\mathsf{Lprof}(.)}$<br>if $\check{k} \neq k$, return 0<br>return 1 |
| Oracle $\mathsf{Lenc}(x)$<br>$\mathsf{PRP}_k(x) \rightsquigarrow (y, \boldsymbol{l}_\mathrm{e})$<br>return $(y, \boldsymbol{l}_\mathrm{e})$ | |

Table 4: Side-channel Key Recovery (KR) experiment.

plaintext is measured $q_\mathsf{a}^\mathsf{r}$ times. As discussed in Section 3.1.2, side-channel key recovery attacks can be of three main flavors: DPA, where both $q_\mathsf{a}^x$ and $q_\mathsf{a}^\mathsf{r}$ are chosen by the adversary; repSPA, where $q_\mathsf{a}^x$ is bounded by design and $q_\mathsf{a}^\mathsf{r}$ is chosen by the adversary; SPA, where both $q_\mathsf{a}^x$ and $q_\mathsf{a}^r$ are bounded by design.

### 4.1.1 Beyond key recovery

Key recovery is a strong adversarial goal and it can be sufficient to achieve simpler objectives to threaten the security of a cryptographic scheme. For example, a (very) weak goal is to achieve a state distinguishing attack for which the experiment is given in Table 5. Here, the adversary has unrestricted access to encryption and decryption oracles with leakage, can choose two plaintexts $x_0, x_1$, receives the leakage trace corresponding to $x_b$ for a (secret) random bit $b$ and has to tell whether $b$ is 0 or 1. Since the adversary can measure the encryption of $x_0$ and $x_1$, she must only find whether $\mathsf{Lenc}(x_b)$ corresponds to $\mathsf{Lenc}(x_0)$ or $\mathsf{Lenc}(x_1)$.

A similar but stronger adversarial goal is the message distinguishing attack depicted in Table 6. It corresponds to the state distinguishing experiment, excepted that the challenge plaintexts $x_0$ and $x_1$ must be fresh (i.e., cannot be inputs to the leaking encryption oracles nor outputs from the leaking decryption oracles).

| $\mathsf{SD}^{\mathsf{L2}}_{\mathsf{PRP}}$ experiment$(n)$ | |
|---|---|
| Initialization$(n)$<br>$k \leftarrow \$$<br>$b \leftarrow \{0,1\}$ | Oracle $\mathsf{Ldec}(y)$<br>$\mathsf{PRP}_k^{-1}(y) \rightsquigarrow (x, \boldsymbol{l}_\mathrm{d})$<br>return $(x, \boldsymbol{l}_\mathrm{d})$ |
| Oracle $\mathsf{Lprof}(.)$ | Finalization<br>$(x_0, x_1) \leftarrow \mathcal{A}$<br>$\check{b} \leftarrow \mathcal{A}^{\mathsf{Lenc}(.),\mathsf{Ldec}(.)}_{\mathsf{Lprof}(.)}\Big(\mathsf{Lenc}(x_b) \setminus y_b\Big)$<br>if $\check{b} \neq b$, return 0<br>return 1 |
| Oracle $\mathsf{Lenc}(x)$<br>$\mathsf{PRP}_k(x) \rightsquigarrow (y, \boldsymbol{l}_\mathrm{e})$<br>return $(y, \boldsymbol{l}_\mathrm{e})$ | |

Table 5: Side-channel State Distinguishing (SD) experiment.

| $\mathsf{MD}^{\mathsf{L2}}_{\mathsf{PRP}}$ experiment$(n)$ | |
|---|---|
| Initialization$(n)$<br>$k \leftarrow \$$<br>$b \leftarrow \{0,1\}$<br>$S \leftarrow \emptyset$ | Oracle $\mathsf{Ldec}(y)$<br>$\mathsf{PRP}_k^{-1}(y) \rightsquigarrow (x, \boldsymbol{l}_\mathrm{d})$<br>$S \leftarrow S \cup x$<br>return $(x, \boldsymbol{l}_\mathrm{d})$ |
| Oracle $\mathsf{Lprof}(.)$ | Finalization<br>$(x_0, x_1) \leftarrow \mathcal{A}$<br>$\check{b} \leftarrow \mathcal{A}^{\mathsf{Lenc}(.),\mathsf{Ldec}(.)}_{\mathsf{Lprof}(.)}\Big(\mathsf{Lenc}(x_b)\Big)$<br>if $x_0 \in S$ or $x_1 \in S$, return 0<br>if $\check{b} \neq b$, return 0<br>return 1 |
| Oracle $\mathsf{Lenc}(x)$<br>$\mathsf{PRP}_k(x) \rightsquigarrow (y, \boldsymbol{l}_\mathrm{e})$<br>$S \leftarrow S \cup x$<br>return $(y, \boldsymbol{l}_\mathrm{e})$ | |

Table 6: Side-channel Message Distinguishing (MD) experiment.

### 4.1.2 Rating the attack's strength

Figure 27 provides a summary of the different attacks introduced so far, highlighting which attack is easier/harder to perform in terms of adversarial means and adversarial goals. The implications regarding adversarial means are consequences of Section 3.1.2. We discuss the implications regarding adversarial goals with Figure 28, which depicts leakage traces of a 10-cycle parallel (128-bit) implementation of the AES. Informally, it puts forward how the exploitability of the leakage samples in a side-channel trace varies in function of the adversary's goals: lighter grey means harder to exploit, dotted means useless for the goal.

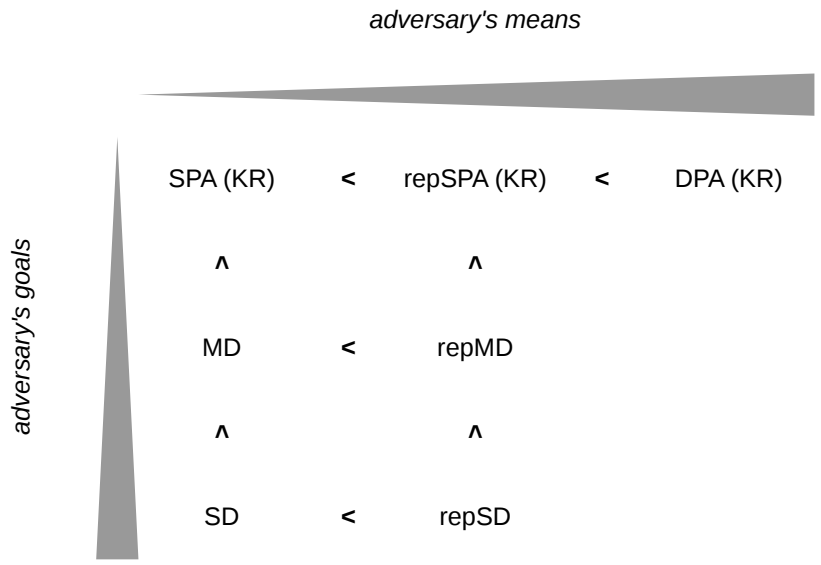
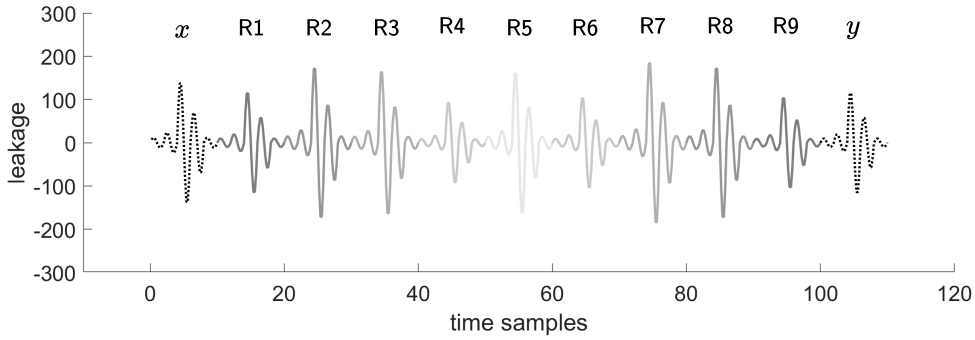
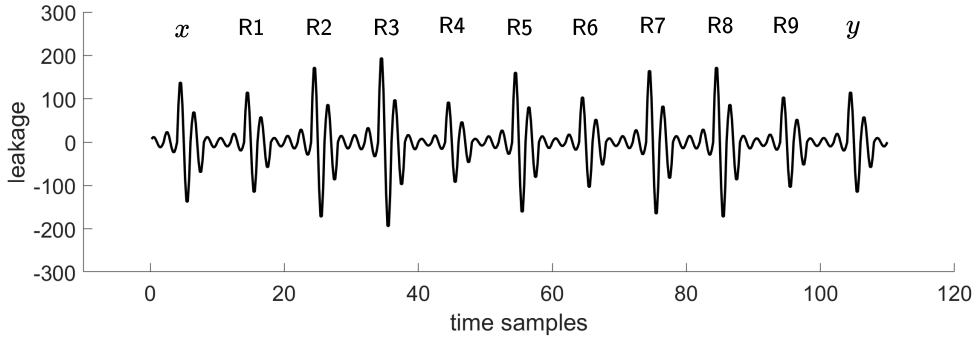



Figure 27: Informal taxonomy of side-channel attacks. Attack A $<$ Attack B means Attack B is easier to perform than Attack A (rep is for variants with repetition).

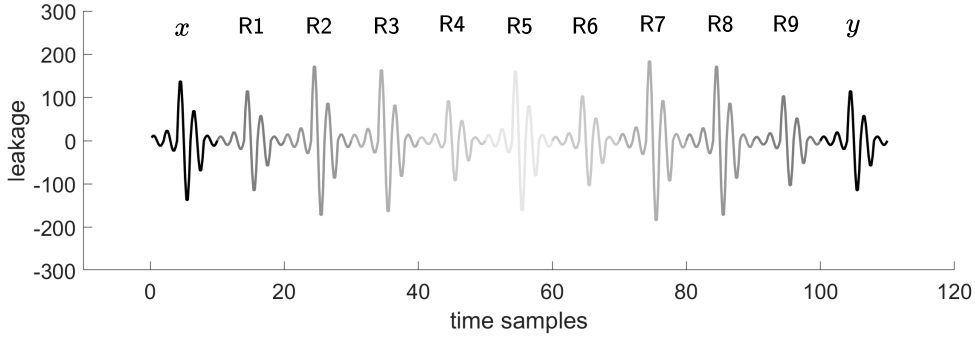
Starting with key recovery attacks, the main observations are twofold. First, leakage on the plaintext $x$ and ciphertext $y$ does not provide information on the key and is therefore dotted. Second, the exploitability of the leakage samples decreases as one digs into the deeper AES rounds. This matches the observation made in the first part of the book that the leakage of external rounds, which may be exploited with divide-and-conquer attacks, is easier to exploit than the leakage of internal rounds, which requires leveraging analytical attacks for which information exploitation is heuristic and optimality cannot be guaranteed.




(a) Key recovery attacks.




(b) State distinguishing attacks.




(c) Message distinguishing attacks.

Figure 28: Leakage exploitability of different attacks (Ri means round $i$).

Following with state distinguishing attacks, we fall into the extreme opposite situation. Since the only thing the adversary has to do in this case is to find out whether a leakage trace $\mathsf{Lenc}(x_b)$ corresponds to $\mathsf{Lenc}(x_0)$ or $\mathsf{Lenc}(x_1)$ for chosen plaintexts $x_0$ and $x_1$, all the leakage samples are trivially exploitable.

Eventually, message distinguishing attacks correspond to an intermediate situation. Since $x_0$ and $x_1$ are now fresh, the adversary cannot directly use all the leakage samples to compare $\mathsf{Lenc}(x_b)$ with $\mathsf{Lenc}(x_0)$ and $\mathsf{Lenc}(x_1)$. Yet, she can still use the plaintext and ciphertext leakage in order to mount a distinguishing attack, and all the other samples in the trace to perform a key recovery.

In the following sections, state distinguishing attacks will be used to discuss what we denote as "impossibility results for efficient countermeasures". Our rationale is that such attacks only require extracting one bit of information from the full traces. Hence, if they have to be prevented for a design, it implies that all its operations must be strongly protected with expensive implementation-level countermeasures (i.e., effectively, be leak-free), so that even this single bit cannot be extracted from the full traces. This contradicts the very goal of mode-level countermeasures, which is to limit the side-channel adversary's attack vectors and the use of implementation-level countermeasures as much as possible. As for message distinguishing attacks, they will be used to discuss confidentiality with leakage, to capture the (unavoidable) amount of leakage on the plaintext that can only be minimized thanks to hardware-level countermeasures.

(*Exercise 2.1 is suggested in order to further illustrate the previous content.*)

## 4.2 Integrity with leakage

We start by defining Message Authentication Codes (MAC).

**Definition 6** (MAC)**.** *A MAC is a triple of algorithms* $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Ver})$ *such that:*

$\mathsf{Gen}$*: takes as input a security parameter* $n$ *and outputs a key* $k$ *from* $\mathcal{K}$*.*

$\mathsf{Mac}$*: takes as input the key* $k$ *and a message* $m \in \{0,1\}^*$ *and outputs a tag* $t \leftarrow \mathsf{Mac}_k(m)$ *from the tag space* $\mathcal{T}$ *(the algorithm may be randomized).*

$\mathsf{Ver}$*: takes as input a triple* $(k, m, t)$ *and outputs either* $\top$ *(accept) or* $\bot$ *(reject).*

*We require correctness:* $\forall k \in \mathcal{K}, m \in \{0,1\}^*$, $\mathsf{Ver}_k(m, \mathsf{Mac}_k(m)) = \top$.

In order to define the security of a MAC in the presence of leakage, we need to define the leaking variants of its algorithms. In general, a leaking oracle Lalgo returns both the (black box) result of the algorithm and the leakage provided by its implementation. Furthermore, in case a security definition gives access to multiple oracles $O_1, \ldots, O_\omega$, we say that the adversary is $(q_1, \ldots, q_\omega, \theta)$-bounded if she can make at most $q_i$ queries to oracle $O_i$ and runs in time at most $\theta$.

Based on the above, we can formalize the security requirement of a MAC as the fact that no efficient adversary should be able to generate message/tag pair that did not appear in a previous authentication. We next consider the strong version of unforgeability [BN00], where even producing a new tag for a previously authenticated message should be computationally hard:

**Definition 7** (Unforgeability with leakage [BGP+19])**.** *A* $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Ver})$ *with leakage functions* Lmac *and* Lver *is* $(q_\mathsf{p}, q_\mathsf{m}, q_\mathsf{v}, \theta, \epsilon)$*-strongly existentially unforgeable against chosen message & verification attacks with leakage in tag generation & verification (sUF-L2) if for all* $(q_\mathsf{p}, q_\mathsf{m}, q_\mathsf{v}, \theta)$*-bounded adversaries:*

$$\Pr(\mathsf{sUF\text{-}L2} \rightarrow 1) \leqslant \epsilon.$$

*The* sUF-L2 *experiment is defined in Table 7 and* $q_\mathsf{p}, q_\mathsf{m}, q_\mathsf{v}$ *are the number of queries made to the profiling, tag generation and verification oracles, respectively.*

This definition is the natural extension of the standard existential unforgeability definition for MAC [KL14], where tag generation and verification oracles are augmented with a leakage function. It is possible to define a weaker variant where the verification device is not under physical control of the adversary. She then only has access to a leaking tag generation oracle Lmac and the leaking tag verification oracle Lver is replaced by its black box counterpart Ver in Table 7. We denote this variant as strong existential unforgeability against chosen message and verification attacks with leakage in the tag generation only (sUF-L1).

## 4.3 Confidentiality with leakage

We start by defining (nonce-based) private-key encryption schemes (ENC).

**Definition 8** (Private-key encryption)**.** *A nonce-based private-key encryption scheme is a triple of algorithms* $\mathsf{ENC} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

Gen*: takes as input a security parameter* $n$ *and outputs a key* $k$ *from* $\mathcal{K}$*.*

| sUF-L2 experiment$(n)$ | |
|---|---|
| Initialization$(n)$<br>$k \leftarrow \$$<br>$S \leftarrow \emptyset$ | Oracle Lver$(m,t)$<br>$\mathsf{Ver}_k(m,t) \rightsquigarrow (\top/\bot, \boldsymbol{l}_{\mathrm{v}})$<br>return $(\top/\bot, \boldsymbol{l}_{\mathrm{v}})$ |
| Oracle Lprof$(.)$ | Finalization<br>$(\check{m},\check{t}) \leftarrow \mathcal{A}^{\mathsf{Lmac}(.),\mathsf{Lver}(.)}_{\mathsf{Lprof}(.)}$<br>if $(\check{m},\check{t}) \in S$ return 0<br>if $\mathsf{Ver}(\check{m},\check{t}) = \bot$, return 0<br>return 1 |
| Oracle Lmac$(m)$<br>$\mathsf{Mac}_k(m) \rightsquigarrow (t, \boldsymbol{l}_{\mathrm{m}})$<br>$S \leftarrow S \cup \{(m,t)\}$<br>return $(t, \boldsymbol{l}_{\mathrm{m}})$ | |

Table 7: Strong unforgeability with leakage (sUF-L2) experiment.

Enc*: takes as input the key $k$, a nonce $N \in \mathcal{N}$ and a message $m \in \{0,1\}^*$ and outputs a ciphertext $c \leftarrow \mathsf{Enc}_k(N,m)$ of the same size as (the padded) $m$.*

Dec*: takes as input a triple $(k, N, c)$ and outputs a message $m$.*

*We require correctness:* $\forall k \in \mathcal{K}, N \in \mathcal{N}, m \in \{0,1\}^*$, $\mathsf{Dec}_k(N, \mathsf{Enc}_k(N,m)) = m$.

Using the same notations as for integrity with leakage in the previous section, we can formalize the security of a private-key encryption scheme as the fact that it is hard for an adversary to tell whether a so-called challenge ciphertext corresponds to one or the other message of a fresh pair she generated:

**Definition 9** (CCA security with leakage [GPPS19])**.** *A private-key encryption scheme* $\mathsf{ENC} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage functions* Lenc & Ldec *is* $(q_{\mathsf{p}}, q_{\mathsf{e}}, q_{\mathsf{d}}, \theta, \epsilon)$*-chosen-ciphertext secure with nonce misuse-resilience and leakage in encryption and decryption (CCAmL2) if for all* $(q_{\mathsf{p}}, q_{\mathsf{e}}, q_{\mathsf{d}}, \theta)$*-bounded adversaries:*

$$\left| \mathsf{Pr}\left(\mathsf{CCAmL2} \rightarrow 1\right) - \frac{1}{2} \right| \leqslant \epsilon.$$

*The* CCAmL2 *experiment is defined in Table 8 and* $q_{\mathsf{p}}, q_{\mathsf{e}}, q_{\mathsf{d}}$ *are the number of queries made to the profiling, encryption and decryption oracles, respectively.*

CCAmL2 experiment$(n)$

**Initialization$(n)$**

$k \leftarrow \$$
$b \leftarrow \{0,1\}$
$S \leftarrow \emptyset$

**Profiling queries**

Query $\mathsf{Lprof}(.)$

**Pre-challenge queries**

Query $\mathsf{Lenc}(N, m)$
$\mathsf{Enc}_k(N, m) \rightsquigarrow (c, \boldsymbol{l}_\mathrm{e})$
$S \leftarrow S \cup \{N\}$
return $(c, \boldsymbol{l}_\mathrm{e})$

Query $\mathsf{Ldec}(N, c)$
$\mathsf{Dec}_k(N, c) \rightsquigarrow (m, \boldsymbol{l}_\mathrm{d})$
return $(m, \boldsymbol{l}_\mathrm{d})$

**Challenge query**

Query $\mathsf{Lchal}(N_{ch}, m_0, m_1)$
if $m_0$ and $m_1$ have different length or if $N_{ch} \in S$, abort
$\mathsf{Enc}_k(N_{ch}, m_b) \rightsquigarrow (c_b, \boldsymbol{l}^b_\mathrm{e})$
return $(c_b, \boldsymbol{l}^b_\mathrm{e})$

**Post-challenge queries**

Query $\mathsf{Lenc}(N, m)$
if $N = N_{ch}$ abort
$\mathsf{Enc}_k(N, c) \rightsquigarrow (m, \boldsymbol{l}_\mathrm{d})$
return $(m, \boldsymbol{l}_\mathrm{d})$

Query $\mathsf{Ldec}(N, c)$
if $(N, c) = (N_{ch}, c_b)$ abort
$\mathsf{Dec}_k(N, c) \rightsquigarrow (m, \boldsymbol{l}_\mathrm{d})$
return $(m, \boldsymbol{l}_\mathrm{d})$

Query $\mathsf{Lchaldec}(N_{ch}, m_0, m_1)$
$\mathsf{Dec}_k(N_{ch}, c_b) \rightsquigarrow (m, \boldsymbol{l}^b_\mathrm{d})$
return $\boldsymbol{l}^b_\mathrm{d}$

**Finalization**

$\check{b} \leftarrow \mathcal{A}$(profiling, pre-challenge, challenge and post-challenge queries)
if $\check{b} \neq b$, return 0, else return 1

Table 8: Confidentiality with leakage (CCAmL2) experiment.

We note that this definition puts no restriction on the leakage that the adversary can observe in encryption and decryption: the leakage of the challenge message can be observed, which we denote as leakage-resistance (L); we even

provide the adversary with the decryption leakage of the challenge ciphertext (without returning the corresponding plaintext). This matches applications where users can use some code or content (e.g., firmware updates or on-demand games or movies) but not access to them due to intellectual property restrictions.

By contrast, we only ensure nonce misuse-resilience (m): nonce-misuse is allowed for the pre-challenge and post-challenge queries but we require that the nonce of the challenge query has not been used for any other encryption query [ADL17]. This ensures that confidentiality may be lost for the messages encrypted with misuse but is restored once fresh nonces are used again.

Leakage-resistance can be turned into leakage-resilience (l) by removing the leakage of the challenge query in Table 8. Similarly, misuse-resilience can be turned into misuse-resistance (M) by allowing misuse for this challenge query.

As for the definition of unforgeability with leakage, we can define a CCAmL1 variant where the adversary can only observe leakage in encryption. Variants without the challenge decryption leakage have also been proposed in [GPPS19], which we denote with an additional star (e.g., CCAmL2* or CCAmL1*).

(*Exercise 2.2 is suggested in order to further illustrate the previous content.*)

## 4.4 Composite definitions

We finally move to the case of Authenticated Encryption (AE), which aims to provide both confidentiality and integrity with a single cryptographic scheme.

**Definition 10** (Authenticated encryption)**.** *A nonce-based authenticated encryption scheme is a triple of algorithms* $\mathsf{AE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

Gen*: takes as input a security parameter* $n$ *and outputs a key* $k$ *from* $\mathcal{K}$*.*

Enc*: takes as input the key* $k$*, a nonce* $N \in \mathcal{N}$*, associated data* $A \in \{0,1\}^*$ *and a message* $m \in \{0,1\}^*$ *and outputs a ciphertext* $c \leftarrow \mathsf{Enc}_k(N, A, m)$*.*

Dec*: takes as input a quadruple* $(k, N, A, c)$ *and outputs either a message* $m$ *if the integrity check succeeds or the* $\perp$ *symbol if it fails.*

*As for MAC and private-key encryption, we require correctness:* $\forall k \in \mathcal{K}, N \in \mathcal{N}, A \in \{0,1\}^*, m \in \{0,1\}^*, \mathsf{Dec}_k(N, A, \mathsf{Enc}_k(N, A, m)) = m$*.*

In this definition, the associated data $A$ is authenticated but not encrypted. Concretely the ciphertext typically includes a tag for the integrity check.

Whereas the security of authenticated encryption is usually formalized with an all-in-one definition in the black box setting [RS06], the leakage setting encourages a more composite approach where the analysis of integrity requirements and confidentiality requirements are kept separate. This choice is motivated by the very different physical assumptions that are required to ensure integrity and confidentiality in the presence of leakage. For instance, it is possible to maintain integrity even if some intermediate computations are fully exposed, meaning that implementation-level countermeasures may not be necessary for the corresponding operations. By contrast, confidentiality under leakage cannot tolerate such an unbounded exposure, and thus requires stronger protections.

The integrity requirements of an AE scheme are updated as follows:

**Definition 11** (Ciphertext integrity with leakage [BPPS17])**.** *An authenticated encryption* $\mathsf{AE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage functions* Lenc *and* Ldec *satisfies* $(q_\mathsf{p}, q_\mathsf{e}, q_\mathsf{d}, \theta, \epsilon)$*-ciphertext integrity with nonce misuse-resistance and leakage in encryption end decryption (CIML2) if for all* $(q_\mathsf{p}, q_\mathsf{e}, q_\mathsf{d}, \theta)$*-bounded adversaries:*

$$\mathsf{Pr}(\mathsf{CIML2} \rightarrow 1) \leqslant \epsilon.$$

*The* CIML2 *experiment is defined in Table 9 and* $q_\mathsf{p}, q_\mathsf{e}, q_\mathsf{d}$ *are the number of queries made to the profiling, encryption and decryption oracles, respectively.*

Once again, we can also define a CIML1 variant with leakage in encryption only.

As for the confidentiality requirements of an AE scheme, they are similar to the ones of a private-key encryption scheme. The (CCAmL2, CCAmL1) definitions in Section 4.3 can be used nearly as is, by just updating the semantics of the encryption and decryption oracles in order to incorporate the associated data.

## 4.5 Discussion and related works

We conclude this chapter with a detailed discussion of the rationale supporting the previous definitional framework together with possible extensions.

**Definitions' zoo.** A consequence of composite definitions is that different mixes of integrity and confidentiality requirements can be considered. In this book, we focus on two combinations that have been shown to be practically-relevant:

| CIML2 experiment$(n)$ | |
|---|---|
| Initialization$(n)$<br>$k \leftarrow \$$<br>$S \leftarrow \emptyset$ | Oracle $\mathsf{Ldec}(N, A, c)$<br>$\mathsf{Dec}_k(N, A, c) \rightsquigarrow (m, \boldsymbol{l}_\mathrm{d})$<br>return $(m, \boldsymbol{l}_\mathrm{d})$ |
| Oracle $\mathsf{Lprof}(.)$ | Finalization<br>$(\check{N}, \check{A}, \check{c}) \leftarrow \mathcal{A}^{\mathsf{Lenc}(.),\mathsf{Ldec}(.)}_{\mathsf{Lprof}(.)}$<br>if $(\check{N}, \check{A}, \check{c}) \in S$ return 0<br>if $\mathsf{Dec}(\check{N}, \check{A}, \check{c}) = \perp$, return 0<br>return 1 |
| Oracle $\mathsf{Lenc}(N, A, m)$<br>$\mathsf{Enc}_k(N, A, m) \rightsquigarrow (c, \boldsymbol{l}_\mathrm{e})$<br>$S \leftarrow S \cup \{(N, A, c)\}$<br>return $(c, \boldsymbol{l}_\mathrm{e})$ | |

Table 9: Ciphertext integrity with leakage (CIML2) experiment.

1. *Strong modes.* Two-pass modes aimed to ensure CIML2 and CCAmL2.
2. *Efficient modes.* One-pass modes aimed to ensure CIML2 and CCAmL1.

Many other variants are nevertheless possible besides these two ones. Guo et al. propose a complete characterization in [GPPS19], where plaintext integrity, ciphertext integrity and security against chosen plaintexts and chosen ciphertext attacks are all investigated with leakage in encryption and decryption or encryption only, with or without considering nonce misuse. That means PIL1, PIL2, CIL1, CIL2, PILM1, PILM2, CIML1, CIML2 for integrity notions and CPAL1, CPAL2, CCAL1, CCAL2, CPAmL1, CPAmL2, CCAmL1, CCAmL2 for confidentiality notions.

**Stronger definitions.** One can note that the strongest combination of leakage-resistance and misuse-resistance is not proposed in our framework. This is due to an impossibility of efficient and secure implementation that we now detail.

Let us first recall the requirement of misuse-resistance as the impossibility for an adversary to learn anything from ciphertexts computed with nonce misuse but the fact that they correspond to exactly the same messages [RS06]. Concretely, misuse-resistance is obtained thanks to two-pass designs like the one of Figure 29,

where $m(i)$ denotes the $i$th block of the message $m$. As soon as two messages (denoted as $m$ and $m'$ on the figure) differ in at least one bit, the Initialization Vector (IV) generated by the first pass is different and, therefore, the keystream that is used to encrypt the messages in the second pass differs as well.

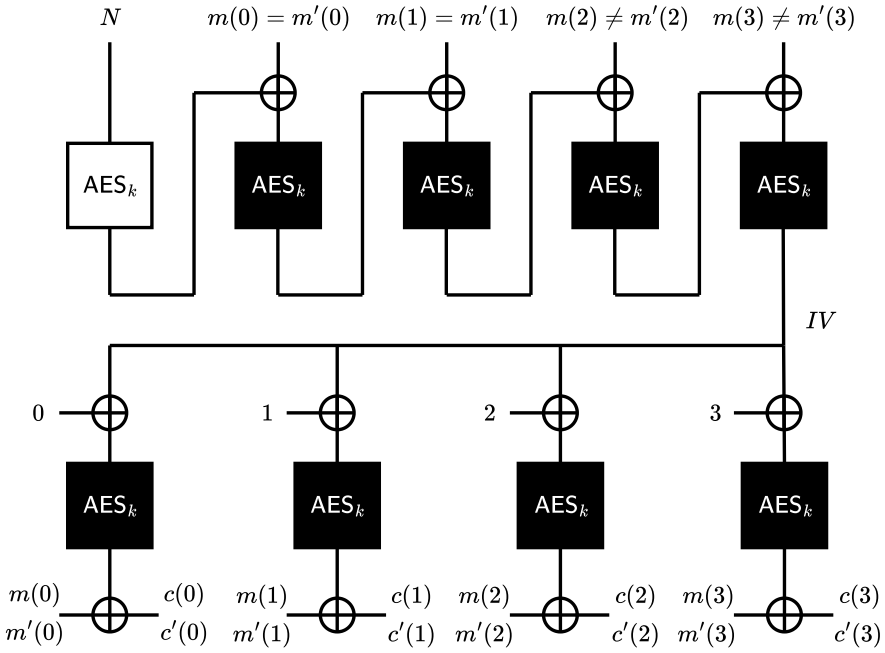



Figure 29: Misuse-resistant authenticated encryption (● means SD attack path).

Let us now consider a leaking implementation of this mode, where the two messages $m$ and $m'$ are identical for their two first blocks and differ afterwards, as depicted on the figure. Misuse-resistance implies that adversaries should not be able to tell whether these first two blocks are identical or not. Yet, if a side-channel adversary is able to detect that their leakage is similar and the leakage of the following blocks differs, she can break this requirement. Concretely, and as reflected by the black color of the AES blocks on the figure, it means that ensuring misuse-resistance with leakage-resistance requires that all the AES implementations must be secure against the state distinguishing attacks of Section 4.1.1. While this is theoretically achievable, it implies that all the AES blocks must be strongly protected thanks to implementation-level countermeasures. In other

words, it prevents mode-level countermeasures that aim to leverage a finer-grain taxonomy of attacks, so that some AES blocks only need to be secure against DPA, SPA or (for integrity requirement) can even leak their operands in full.

Note that despite CCA security with leakage-resistance and misuse-resistance (that would be denoted CCAML2) cannot lead to efficient implementations, one can efficiently combine CCA security with leakage-resistance and misuse-resilience (i.e., CCAmL2) and add misuse-resistance (without leakage) on top [GPPS19].

(*Exercise 2.3 is suggested in order to further illustrate the previous content.*)

**Definitions' style.** Our definitions of confidentiality are based on the left-or-right paradigm [BDJR97]: adversaries must distinguish a left experiment where some message is encrypted from a right experiment where another message is encrypted. We favor this approach over the real-or-random one, where adversaries must distinguish a real encryption algorithm from an idealized function returning random values in the ciphertext space, for two main reasons. First, idealized functions have no implementation, meaning that it is hard to turn them into falsifiable physical assumptions. Second, they create a level of independence between the function's inputs and the outputs which is precisely what side-channel attacks threaten. Therefore, left-or-right definitions appear more attractive in the context of leakage, especially when aiming for standard proofs without relying on idealized assumptions, which is in general an important research goal.

**Separation result.** In the black box setting, the combination of ciphertext integrity and security against chosen plaintext attacks implies CCA security [KY00]. It is shown in [GPPS19] that this implication does not hold with leakage, mainly because it does not capture the risks of side-channel attacks targeting the decryption. Therefore, the combination of CIML2 and CCAmL2 is the strongest possible one given the aforementioned impossibility of efficient modes ensuring CCA security with leakage-resistance and misuse-resistance (i.e., CCAML2). Note that this separation result even holds for the starred definition variants mentioned in Section 4.3, that do not include the challenge decryption leakage.

**Multi-user & multi-challenge variants.** The definitions given in this chapter are in the single-user setting and the confidentiality notions are in the single-challenge setting. There are natural extensions of them, modeling (realistic) threats where an adversary can see the execution of the scheme for different users [BGP$^+$20] and/or several challenge ciphertexts in a single experiment [GPPS19].

**IV-based variants.** We consider nonce-based private-key encryption and AE in this chapter, due to their popularity to reduce the randomness requirements of

symmetric cryptography. Yet, it is worth noticing that IV-based encryption can also be interesting in the context of leakage-resistant cryptography [BPS19].

**Related works.** State distinguishing attacks are rarely discussed in symmetric cryptography. However, they recently gained traction due to their interest for post-quantum cryptographic algorithms relying on the Fujisaki-Okamoto (FO) transform, in order to build public-key encryption schemes secure against chosen ciphertext attacks from public key encryption schemes only secure against chosen plaintext attacks. The FO transform checks the validity of ciphertexts thanks to re-encryption and several pieces of work showed that distinguishing re-encrypted plaintexts based on their leakage can be used to mount powerful (key recovery) attacks that are extremely hard to prevent [RRCB20, XPR+22, UXT+22, ABH+22]. Message distinguishing attacks are rarely discussed as well, as the majority of the side-channel literature focuses on the security of long-term secrets against key recovery. A definition of security for leakage-resilient MAC can also be found in [MOSW15]. Ciphertext integrity with leakage was first introduced without considering decryption leakage [BKP+18]. An alternative definition of authenticated encryption in the presence of leakage can be found in [BMOS17]: it considers a mix of leakage-resilience (i.e., excluding the leakage of the challenge message from the adversary's view) and misuse-resistance rather than a mix of leakage-resistance and misuse-resilience.

# Chapter 5

# Mode-level countermeasures

Given the definitions of the previous chapter, we now focus on how to design authentication, encryption and AE schemes that offer good features for secure implementation. These modes aim to minimize the use of long-term secrets that are natural DPA targets. In Section 5.4, we additionally describe primitive-level solutions turning DPA security requirements into SPA security requirements.

## 5.1 Leakage-resilient MAC

In this section, we detail what can go wrong with MAC constructions that are not designed with leakage in mind and present leakage-resilient variants with the abstractions needed to prove their security in the presence of leakage.

### 5.1.1 What can go wrong?

Let us consider the example of the counter-based XOR-MAC depicted in Figure 30, of which the black box security analysis is given in [BGR95]. An adversary trying to break the sUF-L2 game of Table 7 can leverage the following two strategies:

1. Since all the AES blocks in the figure use the same long-term key $k$, she can perform a DPA on these AES blocks using the $\mathsf{Lmac}$ and $\mathsf{Lver}$ oracles.

2. She can also query the leaking verification oracle $\mathsf{Lver}(\check{m}, t^*)$ for a fresh message $\check{m}$ and candidate tags $t^*$ and try to recover the correct tag $t$ thanks to multiple leaking comparisons between $t$ and $t^*$ [BMPS21].

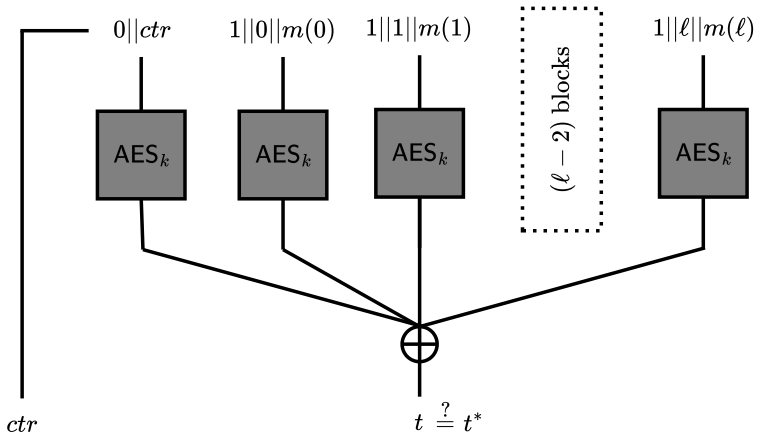



Figure 30: Counter-based XOR-MAC (● means DPA attack path).

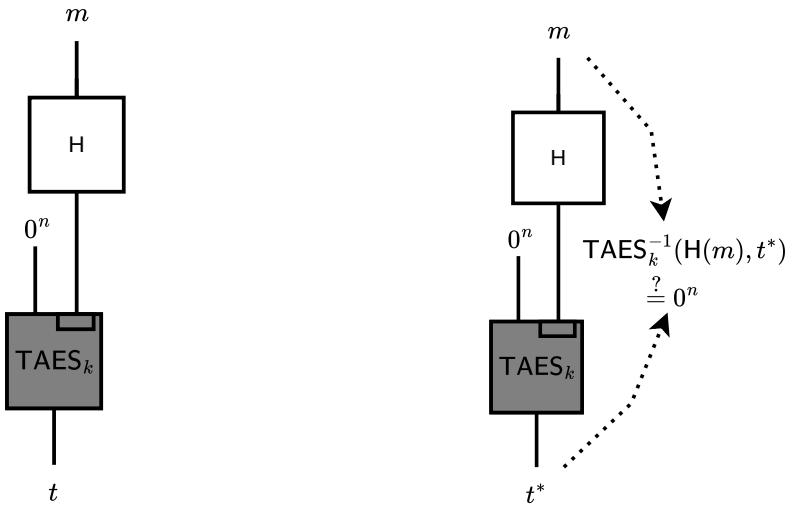
The first attack is standard. The second one just leverages the fact that the verification oracle recomputes the correct tag $t$ and XORs it with the candidate $t^*$ chosen by the adversary during the comparison, leading to a DPA attack path.

### 5.1.2 Design principles

We now detail how to make the aforementioned attack paths more difficult. Being in the symmetric cryptography setting, with $k$ a static parameter shared by the Mac and Ver oracles, we first observe that there is no easy way to update $k$ between different calls to these oracles, as one could leverage in the asymmetric setting [MOSW15]. As a result, executing at least one AES call with strong (implementation-level) DPA countermeasures seems unavoidable. The main goal of symmetric leakage-resilient MAC is to avoid implementing the entire tag generation and verification algorithms with such strongly protected blocks.

One solution for this purpose is to first process the message with a collision-resistant hash function H, so that a strongly protected (tweakable) block cipher only has to be applied once afterwards. Figure 31a illustrates the LRMAC1 design proposed in [BGPS21] for this purpose. It is easy to see that if the strongly protected (tweakable) block cipher implementation is leak-free, the Lmac oracle and the Lver oracle excluding the tag comparison do not provide any more information than their black box counterparts. We show next that the security of this

construction can be proven (and gracefully degrades) under a weak and falsifiable assumption, namely the unpredictability of the (tweakable) block cipher.




(a) Tag generation. (b) Tag verification.

Figure 31: Leakage-resilient MAC (◯ means unbounded leakage tolerated).

Moving to the second attack strategy, we can also see that LRMAC1 does not bring much improvement if we keep on comparing tags as $\mathsf{Mac}_k(m) \stackrel{?}{=} t^*$: without a strongly protected equality testing component, a DPA against the tag comparison operation remains possible. There is however another way of performing the verification, illustrated in Figure 31b. That is, we may test whether $\mathsf{TAES}_k^-1(\mathsf{H}(m), t^*) \stackrel{?}{=} 0$. The advantage of this variant is that if an adversary provides an incorrect tag $t^*$ and even if the comparison is leaked in full, this adversary only learns $\mathsf{TAES}_k^{-1}(\mathsf{H}(m), t^*) \neq 0$ rather than learning the correct tag.*

(*Exercise 2.4 is suggested in order to further illustrate the previous content.*)

* There are variants of this construction given in [BGP$^+$19, BGPS21], checking whether $\mathsf{TAES}_k^{-1}(h_0, t^*) \neq h_1$ for $\mathsf{H}(m) = h_0||h_1$ or, using only block ciphers, whether $\mathsf{AES}_k^{-1}(t^*) \neq \mathsf{H}(m)$, but they lead to the need of stronger assumptions and worse security bounds.

### 5.1.3 Security analysis

In order to analyze the leakage-resilient MAC of Figure 31, we first need to define the standard property that the hash function H is collision-resistant.

**Definition 12** (Collision-resistance)**.** *A family of hash functions* $\mathsf{H}_s : \mathcal{G} \times \{0,1\}^* \rightarrow \mathcal{H}$ *is* $(\theta, \epsilon)$*-collision resistant if for all* $\theta$*-bounded adversaries and uniformly random* $s$ *values* $\in \mathcal{G}$*, the probability to output a pair of distinct inputs* $m_0 \neq m_1$ *such that* $\mathsf{H}_s(m_0) = \mathsf{H}_s(m_1)$ *is bounded by* $\epsilon$*:*

$$\Pr\left(\mathcal{A}(s) \rightarrow (m_0, m_1) \text{ s.t. } m_0 \neq m_1,\ \mathsf{H}_s(m_0) = \mathsf{H}_s(m_1)\right) \leqslant \epsilon.$$

We also need to formalize that the TBC is strongly protected against DPA. For this purpose, we assume that it is unpredictable in the presence of leakage.

**Definition 13** (Unpredictability with leakage [BGP$^+$19])**.** *A tweakable pseudo-random permutation* $\mathsf{TPRP} : \{0,1\}^{2n} \times \{0,1\}^n \rightarrow \{0,1\}^n$ *with leakage functions* $\mathsf{Lenc}$ *and* $\mathsf{Ldec}$ *is* $(q_\mathsf{p}, q_\mathsf{e}, q_\mathsf{d}, \theta, \epsilon)$*-strongly unpredictable with leakage in encryption and decryption (sUP-L2) if for all* $(q_\mathsf{p}, q_\mathsf{e}, q_\mathsf{d}, \theta)$*-bounded adversaries:*

$$\Pr(\mathsf{sUP\text{-}L2} \rightarrow 1) \leqslant \epsilon.$$

*The* sUP-L2 *experiment is defined in Table 10 and* $q_\mathsf{p}, q_\mathsf{e}, q_\mathsf{d}$ *are the number of queries made to the profiling, encryption and decryption oracles, respectively.*

In practice, assuming unpredictable ciphers is widely believed to be more relaxed than assuming PRPs and it is sometimes suggested that reduced-round algorithms can be sufficient to reach this goal [DS09]. As a consequence, unpredictability with leakage is also a weak physical assumption. Besides, it is falsifiable in the sense that the game of Table 10 directly translates into an experiment that evaluation laboratories can try to break (e.g., by performing a key recovery).

Based on these two (weak) assumptions, the leakage-resilient MAC of Figure 31 can be proven secure in the liberal unbounded leakage model.

**Definition 14** (Unbounded leakage model [BKP$^+$18])**.** *An implementation of a scheme is said to offer a security property in the unbounded leakage model if that property is satisfied even if the leakage yields all the internal states produced during each execution of the scheme, including keys and random coins and excluding the internal state of strongly DPA-protected components if there are any.*

| sUP-L2 experiment$(n)$ | |
|---|---|
| Initialization$(n)$<br>$k \leftarrow \$$<br>$S \leftarrow \emptyset$ | Oracle Ldec$(tw, y)$<br>$\mathsf{TPRP}_k^{-1}(tw, y) \rightsquigarrow (x, \boldsymbol{l}_\mathrm{d})$<br>$S \leftarrow S \cup \{(tw, x, y)\}$<br>return $((x, \boldsymbol{l}_\mathrm{d}))$ |
| Oracle Lprof$(.)$ | Finalization<br>$(\check{x}, \check{tw}, \check{y}) \leftarrow \mathcal{A}^{\mathsf{Lenc}(.),\mathsf{Ldec}(.)}_{\mathsf{Lprof}(.)}$<br>if $(\check{x}, \check{tw}, \check{y}) \in S$ return 0<br>if $\check{y} = \mathsf{TPRP}_k(\check{tw}, \check{x})$<br>return 1 |
| Oracle Lenc$(tw, x)$<br>$\mathsf{TPRP}_k(tw, x) \rightsquigarrow (y, \boldsymbol{l}_\mathrm{e})$<br>$S \leftarrow S \cup \{(tw, x, y)\}$<br>return $(y, \boldsymbol{l}_\mathrm{e})$ | |

Table 10: Strong unpredictability with leakage (sUP-L2) experiment.

**Proposition 2** (Leakage-resilient MAC [BGPS21])**.** *Let* H *be a* $(\theta_1, \epsilon_{\mathsf{cr}})$*-collision resistant hash function. Let* TAES *be* $(q_\mathsf{p}, q_\mathsf{m}, q_\mathsf{v}, \theta_2, \epsilon_{\mathsf{unp}})$*-strongly unpredictable with leakage in encryption and decryption (sUP-L2). Then, the MAC specified in Figure 31 is* $(q_\mathsf{p}, q_\mathsf{m}, q_\mathsf{v}, \theta_{\mathsf{unf}}, \epsilon_{\mathsf{unf}})$*-strongly existentially unforgeable against chosen message and verification attacks with leakage in tag generation and verification (sUF- L2) in the unbounded leakage model and:* $\epsilon_{\mathsf{unf}} \leqslant \epsilon_{\mathsf{cr}} + q_\mathsf{v} \cdot \epsilon_{\mathsf{unp}} + \mathsf{negl}(n)$.

*Proof sketch.* In order to break the sUF-L2 security of LR-MAC1, the adversary has to find a fresh and valid pair $(\tilde{m}, \tilde{t})$. For this, she can either find a collision against the hash function H, or find a fresh and valid tuple $(tw, x, y)$ against the sUP-L2 security of TAES, in one of her $q_V$ verification queries. □

**Concrete instantiation.** With a cryptographic hash function, the $\epsilon_{\mathsf{cr}}$ term is expected to be of birthday type in the tweak space. Hence, a natural idea is to use a TBC with the tweak space larger than the block space, like the TAES2 mentionned in Appendix A.2. Generic constructions like [SS23] can also be considered. In this case, finding a collision requires the computation of around $2^{128}$ hash functions and, therefore, the $q_\mathsf{v} \cdot \epsilon_{\mathsf{unp}}$ term is expected to dominate the bound.

## 5.2 Leakage-resistant encryption

In this section, we detail what can go wrong with encryption schemes that are not designed with leakage in mind and present leakage-resistant variants with the abstractions needed to prove their security in the presence of leakage.

### 5.2.1 What can go wrong?

Let us consider the example of the counter-based encryption depicted in Figure 32, of which the black box security analysis is given in [BDJR97]. An adversary trying to break its confidentiality with leakage can again leverage two strategies:

1. Since all the AES blocks in the figure use the same long-term key $k$, she can perform a DPA on these AES blocks using the Lenc and Ldec oracles.

2. The adversary can also try to exploit the leakage obtained during the encryption of $m_b$ during the challenge query, in order to directly identify whether $m_0$ of $m_1$ is encrypted thanks to a message distinguishing attack.

These two strategies have quite different requirements. The first one is a standard DPA and can leverage the leakage of many different plaintexts. As illustrated in Figure 27, the second one corresponds to a considerably weaker goal but must be carried out with the leakage of a single plaintext (possibly with repetition).

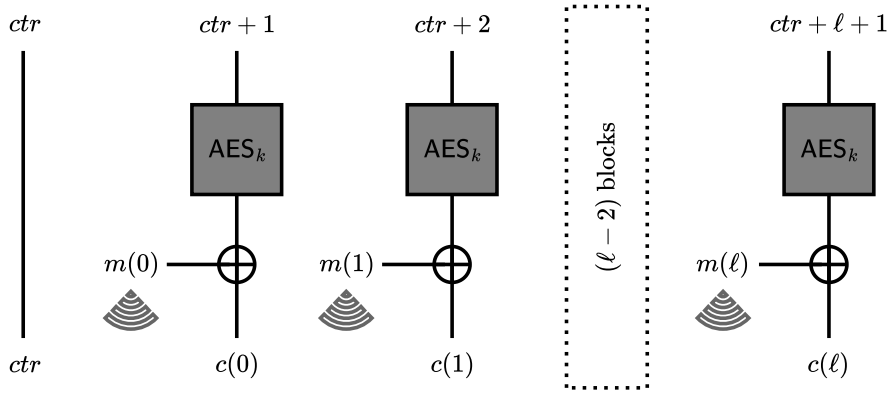



Figure 32: Counter-based private-key encryption.

### 5.2.2 Design principles

We now detail how to make these attack paths more difficult. As in the MAC case, we first aim to minimize the use of the long-term key $k$. Yet, this time, we cannot process the plaintext without secret material: we need a key to hide it.

One solution for this purpose is to start the encryption process with a Key Derivation Function (KDF). For example, one can take a unique nonce $N$ and derive a per-message key $k_0 = \mathsf{AES}_k(N)$. This ephemeral key could then be used to encrypt a message, for example using the counter mode of Figure 32.

Let us momentarily consider security against chosen plaintext attacks with fresh nonces for simplicity (i.e., CPAL1), which is defined by removing the decryption queries from Definition 9. Implementing this solution with a strongly DPA-protected AES implementation for the KDF would be good enough for leakage-resilience instead of leakage-resistance. This is because in this context, the leakage of the challenge plaintext is excluded from the adversary's view. Hence, the mode could loose its confidentiality guarantees in the presence of leakage (e.g., if an adversary performs a DPA against the per-message key), but these guarantees are restored as soon as leakage is again removed from the adversary's view.

For leakage-resistance though, the above solution leaves implementers with the task of protecting all the AES calls of the counter mode with strong DPA countermeasures. Hence, we need an additional ingredient, namely a key evolution scheme. Figure 33 illustrates the design proposed in [PSV15] for this purpose. Its main twist is to derive one ephemeral secret $k_i$ per message block. As a result, a side-channel adversary willing to compromise the confidentiality of the encryption by targeting its ephemeral key material would have to perform a SPA against the per-message-block keys, reflected by the lighter grey color on the figure.

As for the second attack strategy, we observe that the encryption scheme of Figure 33 makes a minimal use of the message blocks (i.e., loading and XORing them with the keystream). What remains is the unavoidable leakage due to this manipulation, which essentially means that ensuring confidentiality with leakage requires preventing the message distinguishing attacks of Section 4.1.1.

### 5.2.3 Security analysis

We next analyze the security of the leakage-resistant private-key encryption scheme of Figure 33. For simplicity, we focus on the key evolution part and assume

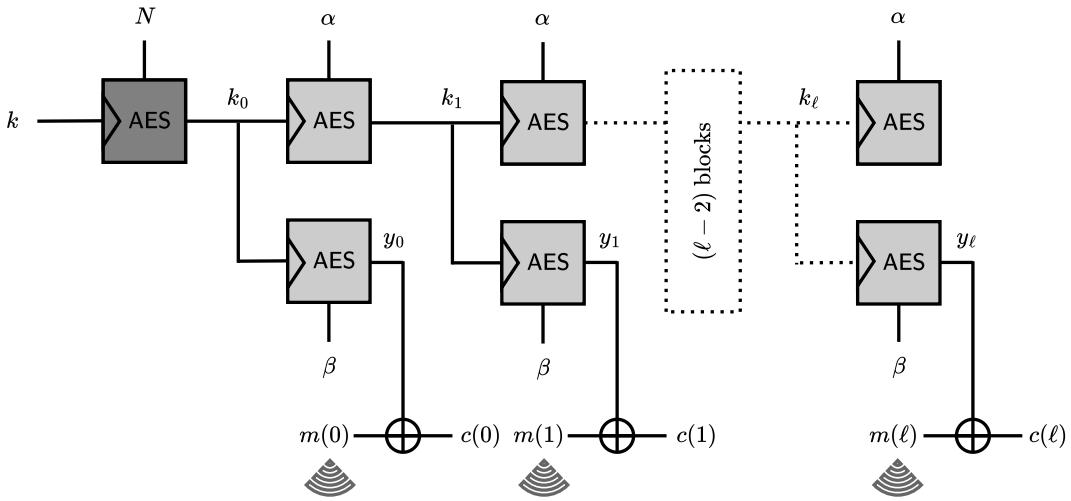



Figure 33: Leakage-resistant private-key encryption (● means SPA attack path).

that the strongly DPA-protected KDF that generates $k_0$ is leak-free. We first need the standard property that the AES is a pseudo-random permutation.

**Definition 15** (Pseudo-random permutation)**.** *A block cipher* $\mathsf{BC} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ *is an* $(\theta, \epsilon)$*-pseudo-random permutation (PRP) if for all* $\theta$*-bounded adversaries, it holds that:*

$$\left| \Pr\left(\mathcal{A}^{\mathsf{BC}_k(.)} \rightarrow 1\right) - \Pr\left(\mathcal{A}^{\mathsf{P}(.)} \rightarrow 1\right) \right| \leqslant \epsilon,$$

*where* $k$ *is a random key in* $\{0,1\}^n$ *and* $\mathsf{P} \in \mathcal{P}$ *is an* $n$*-bit random permutation.*

We also need to formalize that each execution of the AES on a per-message-block key does not leak too much. For this purpose, we rely on the notion of simulatable leakage introduced in [SPY13]. Informally, it requires that it is possible to simulate the leakage of an AES implementation without knowledge of its secrete key, by just manipulating the profiling oracle. This is attractive because it circumvents the difficulty to formalize the informativeness and computational complexity of the leakage function (see the discussion in Section 5.5).

**Definition 16** (Simulatable leakage [SPY13])**.** *A pseudo-random permutation* $\mathsf{PRP} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ *implemented with leakage function* Lenc *has* $(q_{\mathsf{p}}, q_{\mathsf{e}},$

*$\theta_{\mathsf{sim}}, \theta, \epsilon)$-simulatable leakage if there is an efficient $\theta_{\mathsf{sim}}$-bounded simulator such that for all $(q_{\mathsf{p}}, q_{\mathsf{e}}, \theta)$-bounded adversaries:*

$$\left| \Pr\left(\text{q-sim} \rightarrow 1\right) - \frac{1}{2} \right| \leqslant \epsilon.$$

*The* q-sim *experiment is defined in Table 11, where $q_{\mathsf{p}}$ and $q_{\mathsf{e}}$ are the number of queries made to the profiling and encryption oracles, respectively.*

| q-sim experiment$(n)$ | |
|---|---|
| Initialization$(n)$<br>$k, k^* \leftarrow \$$<br>$b \leftarrow \{0,1\}$ | Oracle $\mathsf{Gen}(z, x)$<br>if $b = 0$, return $\mathsf{sim}_{\mathsf{Lprof}(.)}(z, x, k)$<br>return $\mathsf{sim}_{\mathsf{Lprof}(.)}(z, x, k^*)$ |
| Oracle $\mathsf{Lprof}(.)$ | Finalization<br>$(x_0, \ldots, x_{q_{\mathsf{e}}}, z, x) \leftarrow \mathcal{A}$<br>$\check{b} \leftarrow \mathcal{A}^{\mathsf{Lenc}(x_0),\ldots,\mathsf{Lenc}(x_{q_{\mathsf{e}}})}_{\mathsf{Gen}(z,x),\mathsf{Lprof}(.)}$<br>if $\check{b} \neq b$, return 0<br>return 1 |
| Oracle $\mathsf{Lenc}(x)$<br>$\mathsf{PRP}_k(x) \rightsquigarrow (y, \boldsymbol{l}_{\mathrm{e}})$<br>if $b = 0$, return $(y, \boldsymbol{l}_{\mathrm{e}})$<br>return $(y, \mathsf{sim}_{\mathsf{Lprof}(.)}(k^*, x, y))$ | |

Table 11: Simulatable leakage experiment.

Besides the $q_{\mathsf{e}}$ encryption queries, the adversary gets one more chance of winning the game by being able to query the simulator on inputs of her choice via a Gen query, the ciphertext being the real or the random one depending on the bit $b$. It captures the case where the key used in a block cipher is itself a ciphertext from a previous iteration, as for example happens in the scheme of Figure 33.

In practice, the simulator proposed in [SPY13] and described in Table 12 works by "splitting and concatenating" leakage traces generated with a random key so that it starts with the correct plaintext and ends with the correct ciphertext. Such a simulator is expected to prevent attacks trying to exploit the leakage of the

known plaintext and ciphertext (which are the same in the real and simulated traces), constraining the adversary to either mount an SPA or try detecting an inconsistency before and after the split. It is falsifiable in the sense that the game of Table 11 combined with an instance of simulator like the one of Table 12 directly translates into an experiment that evaluation laboratories can try to break.

Note that the $q$-sim experiment of Table 11 relates to security against SPA since it requires to distinguish a fresh trace. Similarly, a $(q, r)$-sim experiment can be defined, where each of the $q_{\mathsf{e}}$ leaking encryption queries can be repeated $r$ times, for contexts where repSPA is possible (e.g., due to nonce misuse).

Under our independence assumption, a typical target is a 10-cycle parallel (128-bit) implementation of the AES, for which leakage traces are depicted in Figure 28 and every cycle provides the adversary with a 128-bit (e.g., Hamming weight) leakage. With such a level of parallelism, both SPA and repSPA should only mildly reduce the entropy of the key. Similarly, detecting a (for now mathematical) inconsistency in the simulated traces is expected to be hard. This is because the vast majority of the adversary's observations will be close to the mean leakage, which leaves her with a large number of possible values per round, namely $\binom{128}{h}$ with $h$ close to 64. As a result, detecting inconsistencies between round $\varrho$ and $\varrho + 1$ should be computationally intensive.* And quite naturally, this should hold even better in case physical noise affects the measurements and only $q$-sim is required, not $(q, r)$-sim. We will see in the next part of the book that the challenge becomes more difficult if the independence assumption is not fulfilled and adversaries can also track physical inconsistencies in the simulated traces.

Based on these assumptions, we can prove the leakage-resistance of the encryption scheme in Figure 33. We again momentarily restrict ourselves to security against chosen plaintext attacks with fresh nonces and leakage in encryption only (i.e., CPAL1). How to reach CCAmL2 will be discussed in the next section.

**Proposition 3** (Leakage-resistant private-key encryption [PSV15])**.** *Let* AES *be a* $(\theta, \epsilon_{\mathsf{prp}})$*-pseudo-random permutation and its implementation have* $(q_{\mathsf{p}}, 2, \theta_{\mathsf{sim}}, \theta, \epsilon_{\mathsf{sim}})$*-simulatable leakage. Then, the encryption scheme specified in Figure 33 and implemented with a leak-free key derivation function is* $(q_{\mathsf{p}}, q_{\mathsf{e}}, \theta_{\mathsf{cpal}}, \epsilon_{\mathsf{cpal}})$*-chosen-plaintext secure with leakage in encryption only (CPAL1) and:*

$$\epsilon_{\mathsf{cpal}} \leqslant 2\ell \cdot (\epsilon_{\mathsf{prp}} + \epsilon_{\mathsf{sim}}) + \ell \cdot \mathsf{Pr}\left(\mathsf{MD}^{\mathsf{L}}_{\mathsf{PRP}}(n) \to 1\right) + \mathsf{negl}(n).$$

* This argument crucially relies on the parallelism of the implementation. For small serial (e.g., software) implementations, the simulatable leakage assumption is not expected to hold directly, implying the need of additional implementation-level countermeasures [GPSG14].

| Split and concatenate simulator$(k^*, x, y)$ |
|---|
| run $\mathsf{AES}_{k^*}(x) \rightsquigarrow (y^*, \boldsymbol{l}_{\mathrm{e}}^1 \| \boldsymbol{l}_{\mathrm{e}}^2)$ |
| compute $x^* = \mathsf{AES}_{k^*}^{-1}(y))$ |
| run $\mathsf{AES}_{k^*}(x^*) \rightsquigarrow (y, \boldsymbol{l}_{\mathrm{e}}^3 \| \boldsymbol{l}_{\mathrm{e}}^4)$ |
| return $\boldsymbol{l}_{\mathrm{sim}} = \boldsymbol{l}_{\mathrm{e}}^1 \| \boldsymbol{l}_{\mathrm{e}}^4$ |

Table 12: Example of leakage simulator.

*Proof sketch.* We begin by considering the first pair of block cipher calls during the encryption. Thanks to the simulatable leakage assumption, the leakages $\mathsf{L}(k_0, \alpha), \mathsf{L}(k_0, \beta)$ can be replaced with simulated ones $\mathsf{L}(k_0^*, \alpha), \mathsf{L}(k_0^*, \beta)$, and the latter are independent of the key $k_0$. Thus, the key $k_0$ becomes perfectly secret and, by the PRP assumption, the outputs $k_1 = \mathsf{AES}_{k_0}(\alpha)$ and $y_0 = \mathsf{AES}_{k_0}(\beta)$ are indistinguishable from two random values $k_1^*$ and $y_0^*$. One then makes another call to the simulatable leakage assumption to produce leakages corresponding to these random values. By iteratively applying this argument, the key stream blocks $y_0, y_1, \ldots$ turn out to be indistinguishable from random ones and the message blocks $m_0, m_1, \ldots$ are thus encrypted by a one-time pad encryption scheme, up to the (irreducible) one-time leakage of these ephemeral values, which is captured by the $\ell$ Message Distinguishing (MD) attacks in the bound. □

**Concrete instantiation.** The scheme of Figure 33 is directly instantiated with the AES. Its bound is of birthday-type, since $\epsilon_{\mathsf{prp}} \approx \frac{\Theta}{2^n}$. It is tight because all the block cipher calls in the figure use the same constants $\alpha$ and $\beta$. Hence, detecting a collision on the values $\mathsf{AES}_{k_0}(\beta)$, $\mathsf{AES}_{k_1}(\beta)$, … breaks the CPA security.

(*Exercises 2.5 & 2.6 are suggested to further illustrate the previous content.*)

## 5.3 Leakage-resistant AE

We finally discuss how to combine leakage-resilient MAC and leakage-resistant encryption designs to offer integrity and confidentiality in a single primitive.

### 5.3.1 A strong (2-pass) mode

The main goal of the previously described leakage-resilient MAC and leakage-resistant encryption schemes is to enable so-called leveled implementations. That is, they are designed in such a way that security guarantees against leakage can be reached without ensuring that all their blocks are secure against DPA.

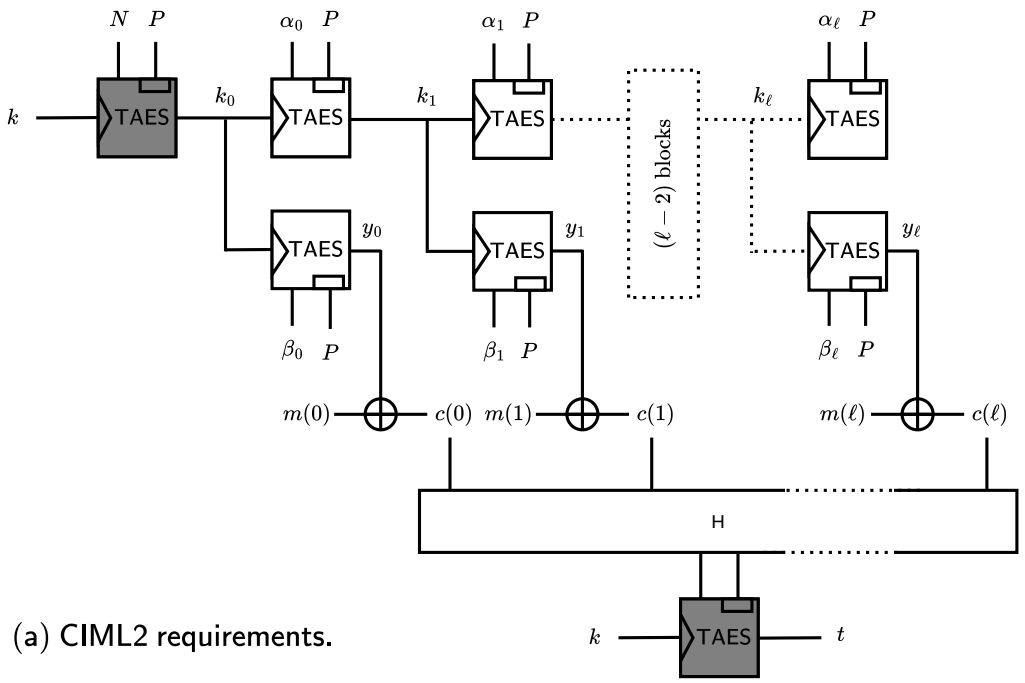
The TEDT mode of operation from [BGP+20] is an example of combined scheme aimed to enable leveled implementations offering the strongest combination of CIML2 and CCAmL2. This scheme essentially applies a leakage-resilient MAC similar to the one of Section 5.1 to ciphertexts produced with an encryption scheme similar to the one of Section 5.2.* As a result, the ciphertexts are only decrypted if the tag verification passes. Its integrity requirements are depicted in Figure 34a. Its confidentiality requirements are depicted in Figure 34b. The main concrete difference with the previous analyzes is that, for the confidentiality guarantees, the adversary can repeat the decryption multiple times and therefore, the blocks processing the message must be secure against repSPA rather than SPA.

The security of TEDT has been analyzed under a leak-free assumption for KDF and the Tag Generation Function (TGF). The scheme offers a tight CIML2 security of $n - \log(n)$ bits in the single-user setting ($n - 2\log(n)$ bits in the multi-user setting). Confidentiality without leakage in the standard model leads to $\approx n/2$ bits of security, which may not be tight since the attack described after Theorem 3 is prevented by the tweaks. Confidentiality without leakage in the ideal cipher model comes with similar bounds as CIML2 security. Confidentiality with leakage falls back to $\approx n/2$ bits of security, due to a powerful attack where detecting collisions thanks to leakage breaks the 2-simulatability assumption (by allowing adversaries to obtain more than two leakage traces per ephemeral key).

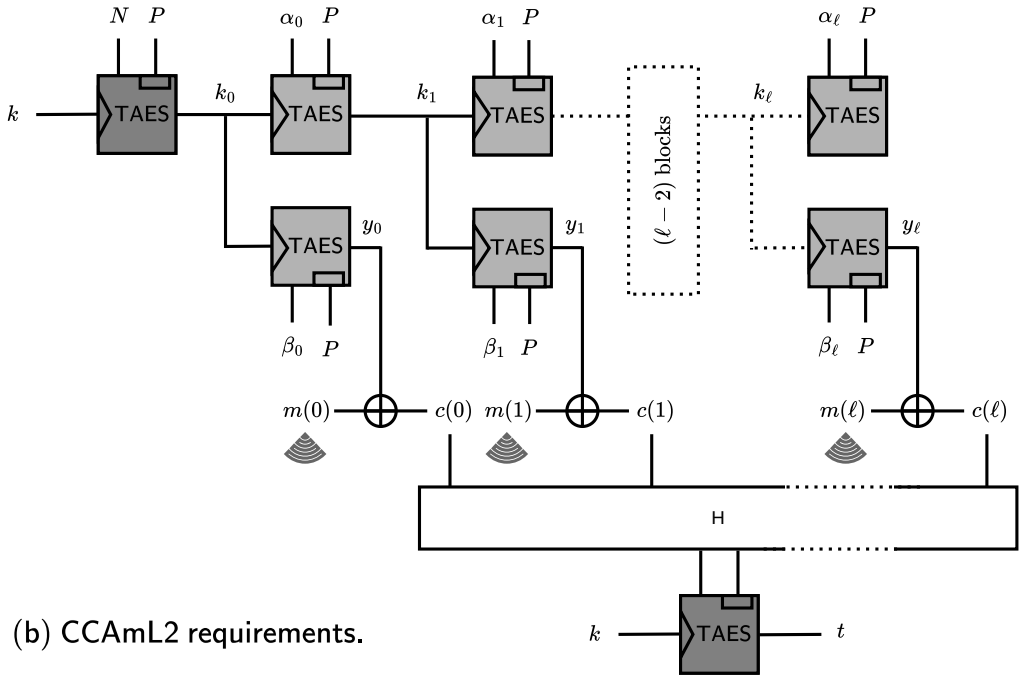
### 5.3.2 An efficient (1-pass) mode

The TET mode of operation from [BGP+20] is an example of combined scheme aimed to enable leveled implementations offering an efficient combination of CIML2 and CCAmL1. It essentially processes the message in a single pass, by combining the keystream generation and the hashing of TEDT in one step. Its integrity requirements are depicted in Figure 35a and are essentially unchanged (also in terms of bounds) compared to TEDT. Its confidentiality requirement are depicted

* The MAC adopts a variant of decryption-based verification used in LRMAC1 (but can be adapted to use the same). The encryption scheme relies on TBCs to obtain better bounds.




(a) CIML2 requirements.




(b) CCAmL2 requirements.

Figure 34: TEDT mode of operation ( means repSPA attack path).

in Figure 35b, with as only change compared to TEDT that CCAmL1 requires AES blocks that are secure against SPA (rather than repSPA) for the message processing part, thanks to the fresh nonce used to encrypt the challenge plaintext guaranteed by the misuse-resilience definition. The security of such a leveled implementation collapses in case leakage is available in decryption. Ensuring CCAmL2 security for TET requires to uniformly protect all of its blocks against DPA.

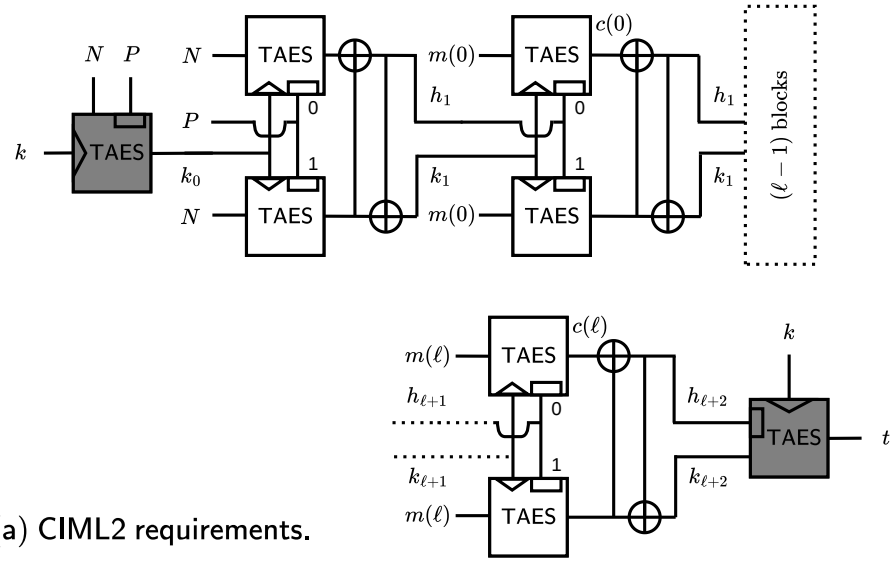
(*Exercise 2.7 is suggested in order to further illustrate the previous content.*)

## 5.4 Getting rid of DPA security

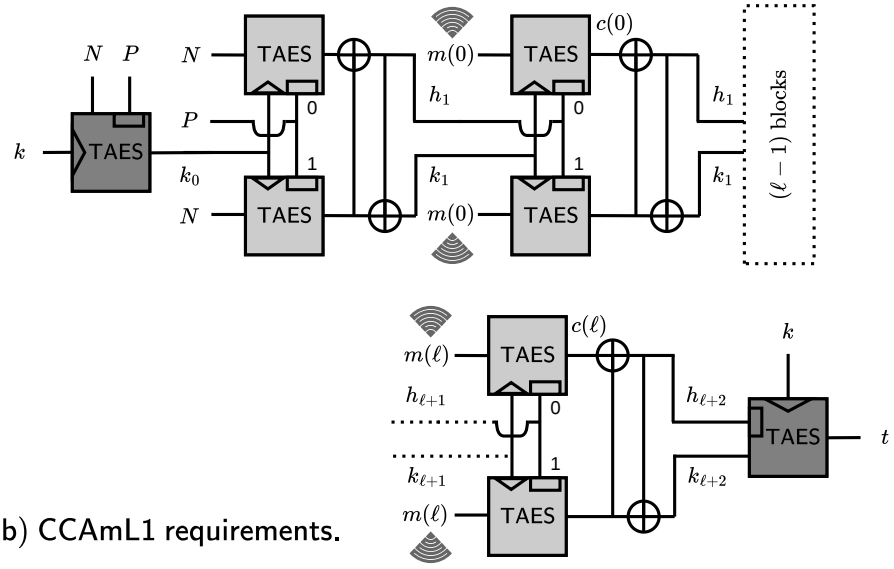
A common pattern of the leakage-resilient MAC of Figure 31, the leakage-resistant encryption scheme of Figure 33 and the TEDT/TET modes of operation of the previous section is that they need at least one execution of a (tweakable) block cipher that is strongly protected against DPA. As detailed in the next chapter, there are implementation-level countermeasures that can be used in order to ensure that this requirement is fulfilled. However, it is also possible to reduce this DPA security requirement to a SPA or repSPA security requirement, by replacing these (T)BC blocks with a leakage-resilient Pseudo-Random Function (PRF).

As illustrated in Figure 36, leakage-resilient PRFs can be viewed as instantiations of the standard (tree-based) PRF construction of Goldreich, Goldwasser and Micali [GGM84]. In their baseline version, they process the input bit by bit, so that every intermediate key $k_i$ of the tree is used at most twice, once to compute $\mathsf{AES}_{k_{i-1}}(\alpha)$, once to compute $\mathsf{AES}_{k_{i-1}}(\beta)$, for public constant values $\alpha$ and $\beta$. As a result, they bound the side-channel adversary to observe only two inputs per intermediate key (i.e., $q_{\mathsf{a}}^x = 2$), at the cost of running the AES $n$ times to process the $n$-bit input $x$. Other tradeoffs between performance and security are obtained by absorbing more than one bit per PRF stage. If $b$ bits are absorbed per stage, $\lceil \frac{n}{b} \rceil$ AES calls are needed to process the $n$ bits of the input $x$, but each intermediate key $k_i$ is used to encrypt $2^b$ public constant values (i.e., $q_{\mathsf{a}}^x = 2^b$).

Leakage-resilient PRFs have been analyzed in different models [DP10, SPY$^+$10, FPS12, YS13, BSH$^+$14] and have also been integrated in leakage-resistant AE modes of operation based on block ciphers [USS$^+$20, BMPS21]. As discussed in [MSV25], and compared to the use of implementation-level countermeasures, the main advantage of such solutions is the simpler implementation they allow (i.e., the fact that they offer a more direct link between functional correctness and security




(a) CIML2 requirements.




(b) CCAmL1 requirements.

Figure 35: TET mode of operation ( means SPA attack path).

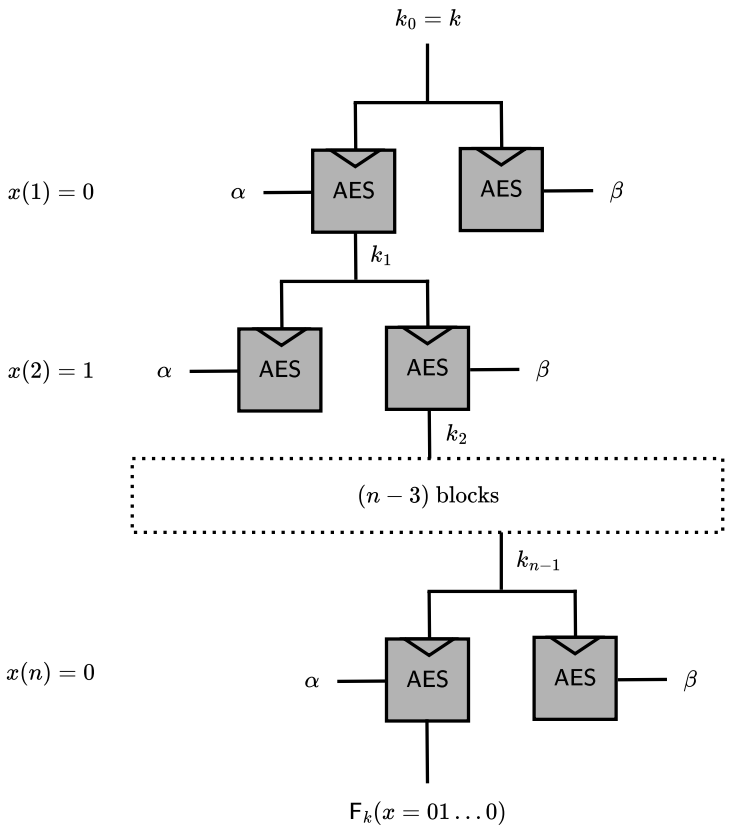



Figure 36: Leakage-resilient PRF (● means repSPA attack path – if used in a KDF for encryption without misuse, it can be limited to an SAP attack path).

against leakage). Their main drawback is to incur overheads even if side-channel attacks are not a concern (since they require a change of primitive).

(*Exercise 2.8 is suggested in order to further illustrate the previous content.*)

## 5.5 Discussion and related works

We conclude this chapter by discussing the implications of the previous results and claims together with alternative leakage models and possible extensions.

**How to interpret the proofs.** It is important to observe that physical security assumptions are of fundamentally different nature than mathematical security assumptions. Mathematical security assumptions are device-independent: they relate to problems that are expected to be hard to solve for any computationally-bounded adversary, given large enough parameters. By contrast, physical security assumptions are device-dependent: the extent to which they hold may be different for different implementations or technologies. From the design point of view, this implies that they essentially correspond to guidelines for implementers. Mode-level countermeasures aim to make these guidelines as clear and easy-to-fulfill as possible, as reflected by the leveled implementations introduced in this chapter. From the evaluation viewpoint, it implies that physical security assumtions must be re-evaluated for each new implementation or technology.

**Other models for confidentiality with leakage.** As first put forward in [MR04], ensuring message confidentiality with leakage is a significantly more challenging task than ensuring message integrity with leakage. One central difficulty in this respect is the requirement, discussed in Section 5.2.3, that the execution of the AES on a single message block should not leak too much. Informally, this implies bounding both the informativeness of the leakage function (i.e., if secrets are leaked in full, no confidentiality is possible) and its computational power. The latter turns out to be highly non-trivial. For example, say the leakage function gives a single bit per execution and is polynomial time – a usual abstraction for efficient functions in modern cryptography. Then, a trivial "future computation attack" against the encryption scheme of Figure 33 is possible: each time an AES key $k_i$ is used, leak one (adaptively chosen) bit of the key $k_{128}$. After the encryption of 128 message blocks, the full $k_{128}$ is recovered. Such an attack is admittedly unrealistic, since it is unlikely that a physical function performs AES computations. Yet, it is allowed by the model because the AES (with known key) can be computed by a polynomial time leakage function. Preventing it formally would therefore require a more appropriate computational restriction of the leakage function. This appears challenging since it needs to solve the dilemma that this function should be simple enough so that it cannot perform cryptographic computations and complex enough so that it can solve Maxwell's equations for complex circuits (which, for example, can take weeks using computer-aided design software).

The simulatable leakage assumption used in Section 5.2.3 is a conceptually elegant way to circumvent the problem: rather than trying to bound the leakage function's informativeness and computational power, it gives the adversary oracle access to this function (i.e., to the leaking implementation). As will be dis-

cussed in the third part of the book, this nevertheless raises additional difficulties when concrete leakages for different operations are not perfectly independent. Meanwhile, we next outline alternative modeling approaches from the literature, highlighting that they all relate to the same (independence) challenge:

*Idealized assumptions.* One simple way to prevent future computation attacks is to assume oracle-free leakage functions [YSPY10]. That is, to assume that the block ciphers of Figure 33 are random oracles that can be queried by the adversary but not the leakage function. Positively, this gives simple intuitions on schemes that have good features for leakage-resistance. Such a computational assumption can also be combined with hard-to-invert leakage [DKL09], which is the weakest possible assumption for bounding the informativeness of the side-channels [FH15]. Negatively, this means assuming independence between the block cipher calls, whereas it is a general goal of cryptographic research to avoid relying on such idealized assumptions (that are not falsifiable). This is even more true when considering physical leakage since, for example, it seems difficult to define the leakage of a random oracle that has no concrete implementation.

*Design tweaks.* Another possibility is to use design tweaks in order to enforce the independence requirement. For example, an "alternating structure", where the key size is doubled and block cipher executions use one or the other half of the key, can be used for this purpose [DP08, Pie09]. This means future computation attacks cannot be performed as long as the leakage function does not provide information about the unused part of the key (i.e., only computation leaks).* Combining fresh but public randomness in the encryption scheme [YSPY10, FPS12], or even pseudo-randomness [YS13], with a non-adaptive leakage function, has also been proposed as an alternative. Such design tweaks are generally combined with the assumption that the leakage function has bounded range or that it preserves the (pseudo-)entropy of the key for the informativeness part, i.e., stronger assumptions for bounding the informativeness of the side-channels [FH15].

**Strengthened black box definitions.** Given the difficulty to bound the computational power of the leakage function discussed above, we note that, strictly, the black box security definitions of collision-resistant hash function (Definition 12) or pseudo-random permutation (Definition 15) to be instantiated in implementations with leakage should also be enhanced with an access to a (profiling) leakage oracle. Indeed, despite unlikely, nothing theoretically prevents the leakage function to perform cryptanalytic work. If the leakage function was polynomial time,

* Which, again, only holds up to some extent in practice [Mor14, PSKM15, MMR20].

these augmented definitions would nevertheless be equivalent to the black box ones. We do not add such an oracle access in our notations for readability.

**Finer-grain analysis.** Eventually, the analyzes in this section are connected to the taxonomy of attacks outlined in Section 4.1.2, and in particular the fact that preventing SPA with implementation-level countermeasure is easier/cheaper than preventing repSPA, which is itself easier/cheaper than preventing DPA. It is however possible to consider a finer-grain taxonomy. For example, in the CBC-MAC and CBC-ENC schemes illustrated in Figure 37, all the encrypted blocks (but the first one) are XORed with an unknown ciphertext. So DPA targeting these block cipher calls requires to deal with unknown inputs in this case, making them marginally more difficult. A similar observation can be made for AE schemes leveraging a whitening of the message blocks with secret values, like OCB [BBB+22].

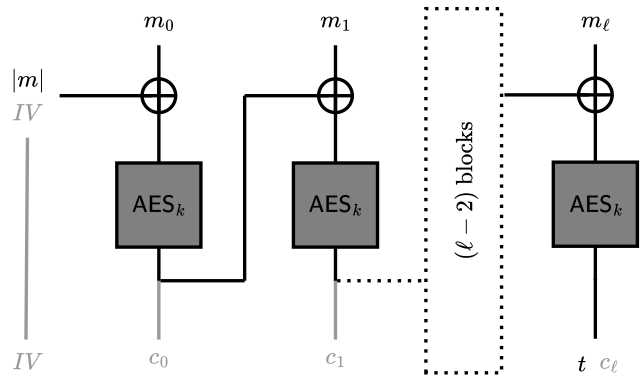



Figure 37: CBC-MAC (in black) and CBC-ENC (grey additions).

**Related works.** An alternative treatment of leakage-resilient symmetric encryption based on re-keying can be found in [ABF13]. Permutation-based designs with good features for security against leakage have been proposed and analyzed in [DM19, DJS19, GPPS20]. The additional challenges raised by the tag verification for permutation-based designs are discussed in [DM21]. Concrete examples of permutation-based leakage-resistant AE include the efficient (1-pass) Ascon [DEMS21] and the strong (2-pass) ISAP [DEM+20]. ISAP additionally relies on a permutation-based leakage-resilient PRF, similar to the one of Section 5.4. A systematic investigation of some leakage-resistant AE according to their mode-level security, relying on the same taxonomy of attacks as in this chapter, can be found in [BBC+20]. Other leakage-resistant schemes relying on tweakable block ciphers include TEDT2 [Lis21], Triplex [SPS+22] and Multiplex [SPS24]. Fork ciphers have been

shown in [DDLM24, BSL24] to be interesting emerging primitives for security against leakage. The leakage-resilient PRF of Section 5.4 and its benefits have also been studied in the chosen input and the unknown input attack settings [MSJ12, MSNF16].

# Chapter 6

# Implementation-level countermeasures

Chapter 5 showed that leakage-resistant modes of operation can hardly get rid of DPA-protected implementations for a few block cipher calls. One option for this purpose is to reduce the DPA security requirements to SPA security requirements, using a leakage-resilient PRF. Yet, this alternative is not always usable (e.g., if standard ciphers and modes have to be used) nor desirable (e.g., if side-channel security is required for only a few devices in a system). Besides, the first part of this book (especially Exercise 1.8) showed that noise addition alone is unlikely to serve as a sufficient security parameter to reach this security goal efficiently. This is because noise addition provides linear security increases at the cost of linear performance overheads. In this section, we therefore investigate implementation-level countermeasures that are natural candidates to ensure DPA security.

## 6.1 Shuffling (operations)

Given the importance of leakage detection in the side-channel analysis workflow, an appealing idea to make DPA more difficult is to randomize the execution of the operations of the implementation to protect. This solution is usually denoted as the shuffling countermeasure, introduced in [HOM06] and carefully analyzed in [VMKS12]. It requires to identify sets of operations that are applied independently, so that they can be executed in randomized order. This is for example feasible in the case of a software (serial) implementation of the SubBytes operation in an AES implementation, which will be our running example.

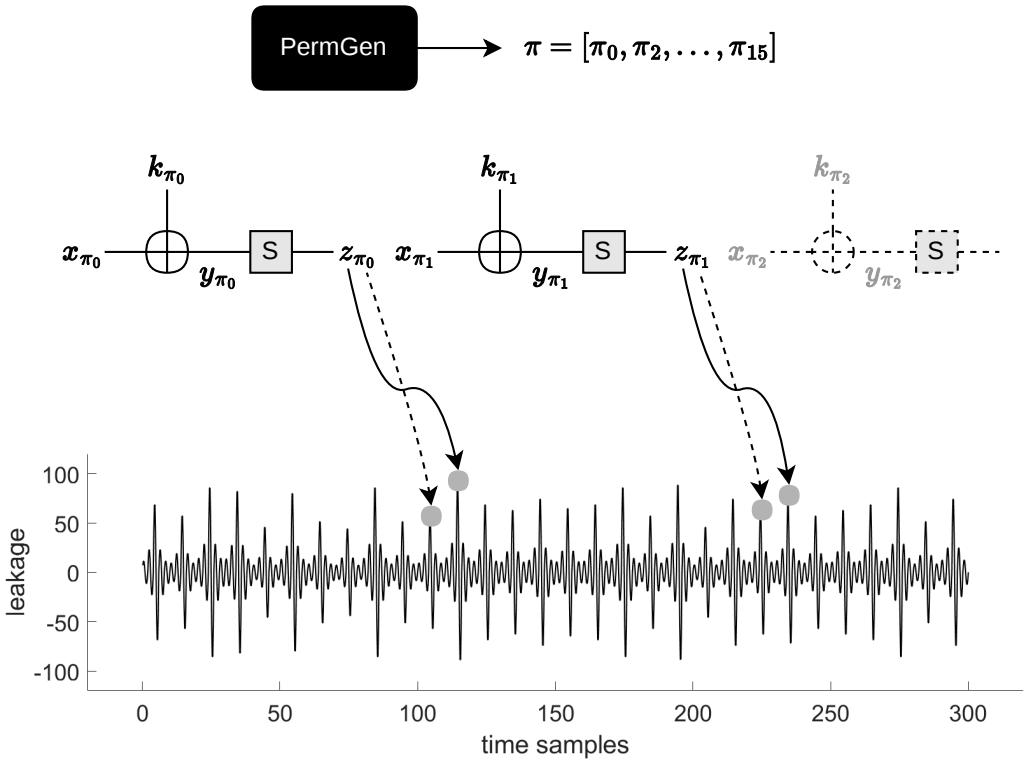



Figure 38: Leakage trace $\boldsymbol{l}$ for shuffled AES S-boxes.

### 6.1.1 High-level modeling

We illustrate the shuffling countermeasure with Figure 38, which is a variant of Figure 1 where the sixteen AES S-boxes are executed according to a random permutation $\pi = [\pi_0, \pi_1, \ldots, \pi_{15}]$. For simplicity, we for now assume that the generation of the permutation is leak-free. In this case, and considering an adversary trying to perform a DPA exploiting the leakage of an S-box output $Z_i$, a vector of two components is available: $\boldsymbol{L} = (\boldsymbol{L}_\pi, \boldsymbol{L}_Z)$. The first one is defined as:

$$\boldsymbol{L}_\pi = [\boldsymbol{L}_{\pi_0}, \boldsymbol{L}_{\pi_1}, \ldots, \boldsymbol{L}_{\pi_{15}}],$$

and we denote it as a direct permutation leakage. As illustrated on the figure, each time an S-box has to be executed in a shuffled implementation, one first has

to load the index of the byte that must be manipulated. The second component is the S-box output leakage obtained from the permuted execution:

$$\boldsymbol{L}_Z = [\boldsymbol{L}_{z_{\pi_0}}, \boldsymbol{L}_{z_{\pi_1}}, \ldots, \boldsymbol{L}_{z_{\pi_{15}}}].$$

Ideally, and in order to assess the security gains provided by the shuffling countermeasure, one would like to compute the mutual information $\mathsf{MI}(Z_i; \boldsymbol{L})$ for each byte $0 \leqslant i \leqslant 15$. Yet, concretely, this requires computing the PDF:

$$\mathsf{f}(\boldsymbol{l}|z_i) = \sum_{\pi \in \mathcal{P}} \mathsf{Pr}(\pi) \cdot \mathsf{f}(\boldsymbol{l}|z_i, \pi). \tag{42}$$

As the set of permutations rapidly increases with the number of shuffled operations (e.g., in our SubBytes case, $16! \approx 2^{44}$), such a direct computation is usually not feasible and simplifying assumptions have to be considered. This means that, even under the (perfect) profiling assumption we leverage in this part of the book, we can only estimate a PI corresponding to tractable adversarial strategies.

We now detail how $\mathsf{PI}(Z_i; \boldsymbol{L})$ can be estimated. We use the improved model from [ABG+22] for this purpose, which corrects a bias of the model from [VMKS12] appearing when the implementation noise increases. As a first step, we need to assign a normalized weight to reflect the probability that a byte $i$ of the AES state is manipulated at cycle $c$ of the shuffled SubBytes execution:

$$w(\pi_c = i|\boldsymbol{l}_{\pi_c}) = \frac{\mathsf{f}(\boldsymbol{l}_{\pi_c}|\pi_c = i)}{\sum_{j=0}^{15} \mathsf{f}(\boldsymbol{l}_{\pi_c}|\pi_c = j)}. \tag{43}$$

Note that $w(\pi_i = i|\boldsymbol{l}_{\pi_i}) = 1$ for unprotected implementation and $w(\pi_c = i|\boldsymbol{l}_{\pi_c})$ is always worth $\frac{1}{16}$ in case there is no direct permutation leakage (i.e., $\boldsymbol{L}_{\pi_c} = \frac{1}{16}$ $\forall\, i \in [0:15]$). We next compute the conditional probabilities for the target byte $z_i$ given the S-box output leakage at each cycle $c$:

$$\mathsf{Pr}(z_i|\boldsymbol{l}_{z_{\pi_c}}) = \frac{\mathsf{f}(\boldsymbol{l}_{z_{\pi_c}}|z_i)}{\sum_{z_i'=0}^{255} \mathsf{f}(\boldsymbol{l}_{z_{\pi_c}}|z_i')}. \tag{44}$$

Thanks to Equations 43 and 44, an estimated model can be derived.* Namely:

$$\hat{\mathsf{Pr}}(z_i|\boldsymbol{l}) = \sum_{c=0}^{15} w(\pi_c = i|\boldsymbol{l}_{\pi_c}) \cdot \mathsf{Pr}(z_i|\boldsymbol{l}_{z_{\pi_c}}), \tag{45}$$

* It slightly differs from the one in [ABG+22] which takes a product over the key bytes to capture the leakage of the entire state whereas we focus on the leakage of a single byte.

from which we can directly obtain the following PI estimation:

$$\tilde{\mathsf{PI}}(Z_i; \boldsymbol{L}) = \mathsf{H}(Z_i) + \sum_{z_i=0}^{255} \mathsf{Pr}(z_i) \cdot \sum_{\boldsymbol{l} \in \mathcal{L}_{\mathsf{t}}(z_i)} \frac{1}{|\mathcal{L}_{\mathsf{t}}(z_i)|} \cdot \log_2 \hat{\mathsf{Pr}}(z_i|\boldsymbol{l}). \quad (46)$$

### 6.1.2 Information theoretic analysis

We assess the impact of the shuffling countermeasure with an information theoretic analysis. For simplicity, we assume that all the elements of the vectors $\boldsymbol{L}_\pi$ and $\boldsymbol{L}_Z$ give rise to Hamming weight leakage with additive Gaussian noise. Based on the notations of Chapter 2, this means $\delta(.) = \mathsf{HW}(.)$ and $r$ is Gaussian-distributed. Given this choice, $\tilde{\mathsf{PI}}(Z_i|\boldsymbol{L})$ can estimated based on Equation 46, using the SNR as a (normalized) metric reflecting the amount of additive noise. The result of such an information theoretic evaluation is in Figure 39.

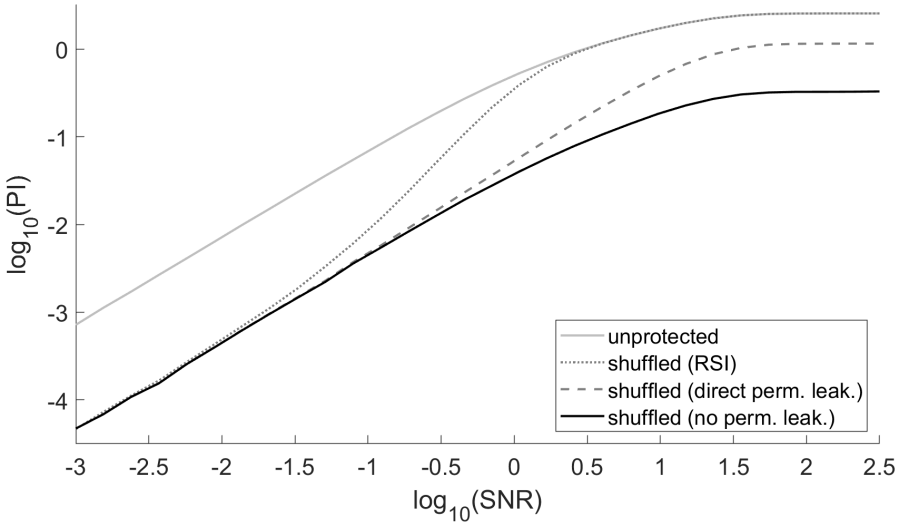



Figure 39: Information theoretic analysis of the shuffling countermeasure.

The figure leads to three main intuitive observations that we detail next.

First, comparing the curve of an unprotected implementations with the one of a shuffled implementation with no permutation leakage, the information (hence,

the wort-case security level) differ(s) by a factor 16. This matches the intuition given for the weights of Equation 43, which are worth 1 vs. $\frac{1}{16}$ in these cases.

Second, this factor 16 is not directly observed when a direct permutation leakage is exploited and the noise level is low (i.e., the SNR is high). This is because the leakage exploited by the adversary then significantly biases the distribution of the permutation $\pi$.* By contrast, as the noise increases (i.e., the SNR decreases), the curve corresponding to a direct permutation leakage gets close to the one without permutation leakage. This is because the information leaked on the permutation then gradually becomes insufficient to accurately assign the manipulation of a key byte to a given cycle. In this respect, it is worth noting that the leakage on the permutation is of SPA type (since it is an ephemeral secret), which is in contrast with the leakage on $K$ obtained via the leakage of $Z$, which is of DPA type.

Third, the situation gets worse if the uniformly random permutation of the shuffling is replaced by a Random Start Index (RSI). In this case, the index of the first executed S-box is randomized and the others follow in order, which reduces the set of permutations from $16! \approx 2^{44}$ to only 16. As a result, the computationally efficient but heuristic model of Equation 45 can be replaced by the (optimal) exhaustive strategy of Equation 42. The combination of such an optimal strategy with the fact that the adversary gets 16 Hamming weights to revover a single 4-bit value even implies that the RSI countermeasure brings no security at high SNR.

### 6.1.3 Discussion and related works

The results in this section show that the shuffling countermeasure is effective, yet not at the level that is expected to prevent DPA from the cryptographic viewpoint. Namely, it can only be used to emulate a constant amount of additional noise in an implementation, at the cost of constant overheads. So just like physical noise, this cannot create a large (ideally exponential vs. polynomial) gap between what adversaries and legit users can do with a leaking implementation. Furthermore, whenever adversaries take advantage of permutation leakage, the information theoretic analysis of Figure 39 shows that this noise emulation only takes place if there is already some physical noise in the measurements. Eventually, the security parameter of the shuffling countermeasure (i.e., the size of the permutation) is quite algorithm-dependent. For example, it applies reasonably well to the SubBytes operation, which executes 16 S-boxes independently. But applying it to the 4 MixColumns operation of the AES only leads to $4! = 24$ permutations, enabling

* Recall that the strategy based on Equations 43, 44 and 45 is possibly suboptimal.

the optimal strategy of Equation 42 and implying larger noise requirements. Besides, running dummy MixColumns operations to increase the set of independent operations that can be shuffled also leads to non-trivial challenges (see the discussion of random delay interrupts in the next related works paragraph).

So overall, shuffling is a deceptive countermeasure against DPA and is inherently more suited to prevent SPA. This is because a constant information reduction can be sufficient to bound the leakage to a satisfactory level (e.g., in an encryption scheme like the one of Figure 33). For the more challenging DPA security requirement, we will see in the next section that randomizing the data leads to a better deal, leading to noise amplification rather than noise emulation.

(*Exercises 2.9 & 2.10 are suggested to further illustrate the previous content.*)

**Related works.** How to generate the shuffling permutation is discussed in [VMKS12]. Attacks that directly target the permutation generation for now remain under-discussed in the literature. As an alternative to shuffling operations, it has been proposed to include random delays within cryptographic implementations [CCD00]. The security of such a proposal scales according to pretty similar intuitions as the shuffling countermeasure, in function of the variance of the delays [Man04, CK09, CK10]. However, for random delays to be effective, the dummy operations used for randomization must remain hard to distinguish from the actual algorithm execution, leading to an additional attack path compared to what the shuffling countermeasure allows [DRS+12]. A similar issue pops up when implementers try to increase the amount of operations to shuffle with dummy operations (e.g., in the MixColumns example mentioned above). Shuffling has recently gained interest in the context of post-quantum cryptography, where it is a popular solution to prevent SPA attacks against leaking implementations of the Number Theoretic Transform (NTT) that is used in many algorithms [PPM17, PP19, RPBC20, HSST23].

## 6.2 Masking (data)

In this section, we introduce the masking countermeasure that is the cornerstone which can turn cryptographic implementations of standard primitives (like the AES) into leakage-resilient ones. Masking has been independently introduced by Chari et al. [CJRR99] and by Goubin & Patarin [GP99], soon after Kocher et al.'s first DPA paper. As an implementation-level countermeasure, it is applicable to any cryptographic algorithm, yet at non-negligible implementation cost. Its secure instantiation raises multiple challenges spanning theory and practice. Given the breadth of the abstractions needed to comprehend masking in depth, we will

split its treatment is different parts. In this first section, we introduce the foundational principles that make it a sound countermeasure from the cryptographic viewpoint. We also exhibit why a formal treatment is needed to take advantage of these principles effectively. Next, in Chapter 7, we dig into the theory of masking under the simplifying (profiling, uncertainty and independence) assumptions that we use for this second part of the book. Finally, the third part of the book will tackle the deviations from these simplifying assumptions, their impact and how to keep them under sufficient control to secure implementations concretely.

### 6.2.1 Convolution intuition

The high-level idea of masking is to encode every sensitive variable in an implementation as a $d$-tuple of shares. Given the choice of the AES to illustrate our discussions, the most natural sharing is the Boolean one. Taking again an S-box output byte as case study, this means that $Z$ is represented by a set of $d$ shares $\{Z_0, Z_1, \ldots, Z_{d-1}\}$ such that their sum equals the correct $Z$, namely:

$$Z = Z_0 \oplus Z_1 \oplus \cdots \oplus Z_{d-1}, \tag{47}$$

where the first $d-1$ shares are picked up uniformly at random and the last one is computed so that they sum to the correct $Z$. Note that in the rest of this chapter, we use the subscript $i$ to denote the shares' indexes rather than the AES byte indexes. This allows lighter notations without loss of needed generality, since we introduce masking for elementary operations. Besides, we start by considering the security of such an encoding in isolation. How to compute securely on secret shared data is discussed in Section 6.3, Chapter 7 and Part III of the book.

Interestingly, we already have all the necessary background to understand why masking offers a significant security increase compared to shuffling. Indeed, it can be viewed as a direct application of the SASCA in Section 2.4.1 and their convolution-based optimization in Section 2.4.2. For this purpose, we start by illustrating the leakage of an encoding in Figure 40. Similarly to the shuffling case, we consider an adversary having access to a leakage vector:

$$\boldsymbol{L} = [\boldsymbol{L}_0, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_{d-1}]. \tag{48}$$

Let us first consider a 2-share example. The adversary then obtains a leakage $\boldsymbol{l}_0$ on $z_0$, a leakage $\boldsymbol{l}_1$ on $z_1$ and wants to gain information about their XOR, since $z = z_0 \oplus z_1$. But this is precisely what SASCA do when updating factor-to-variable

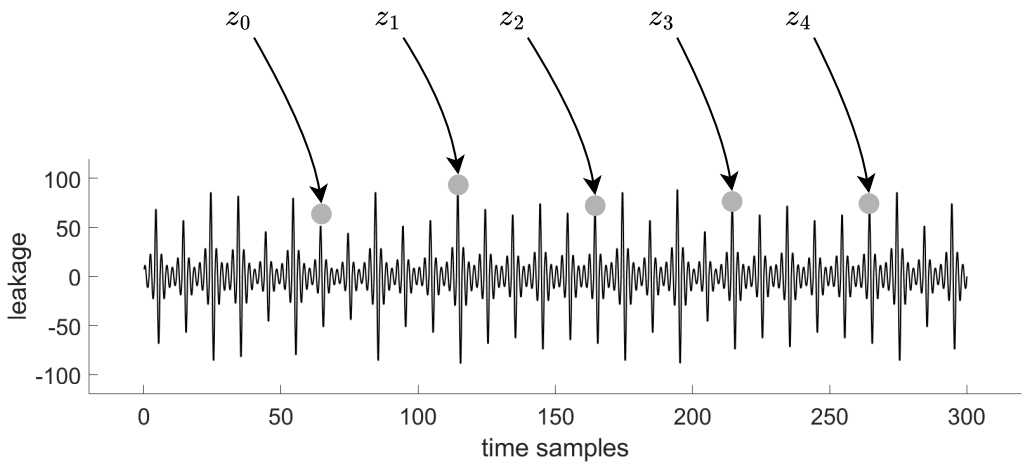



Figure 40: Leakage trace $\boldsymbol{l}$ for masked encoding with $d = 5$ shares.

messages for a XOR. So, turning the leakage on share 0 into a vector of probabilities $\hat{\boldsymbol{p}}_{z_0}$ and the leakage on share 1 into a vector of probabilities $\hat{\boldsymbol{p}}_{z_1}$, information on $z$ is obtained by computing $\hat{\boldsymbol{p}}_z = \hat{\boldsymbol{p}}_{z_0} * \hat{\boldsymbol{p}}_{z_1}$.* This directly generalizes to more shares, so that the Probability Mass Function (PMF) of the secret is obtained by performing convolution products on the PMF of the individual shares. Noise amplification then derives from Young's convolution inequality [You12], which essentially shows that the convolution product is a blurring operation.

It turns out the intuition is even simpler when using the LRPM of Section 3.2.2. In this case, denoting the (normalized) mutual information on each share $Z_i$ as $\lambda_i$, the normalized mutual information on $Z$ is given by Equation 37 as:

$$\lambda(Z) = \prod_{i=0}^{d-1} \lambda_i. \tag{49}$$

From the above, one can see that, at least in the low SNR regimes where the LRPM is expected to be meaningful, masking amplifies the uncertainty (e.g., noise) that remains on the shares. Assuming that each share leaks approximately the same amount of information, we now have that information decreases (hence, security

* In the simplified case considered here, where the encoding is considered in isolation, SASCA even converges to exact probabilities since the factor graph is a tree and BP is exact.

grows) exponentially in $d$. So for the masking countermeasure to be cryptographically sound, it only remains to show that its overheads do not scale exponentially as well – which we will discuss carefully in Section 6.3 and Chapter 7.

### 6.2.2 Statistical intuition

Besides the convolution intuition in the previous section, the exponential security amplification of the masking countermeasure can also be expressed based on a statistical argument. For this purpose, let us again start with a 2-share example: $z = z_0 \oplus z_1$. Let us further assume that $z$, $z_0$ and $z_1$ are 1-bit values for simplicity. Let us finally assume a noisy Hamming weight leakage function:

$$l_i = \mathsf{HW}(z_i) + r_i, \tag{50}$$

with $r_i$ a Gaussian-distributed noise. As in the shuffling case, we are interested in the estimation of the leakage PDF conditioned on $z$.* That is:

$$\mathsf{f}(\boldsymbol{l}|z) = \sum_{z_0=1}^{255} \cdot \sum_{z_1=1}^{255} \ldots \sum_{z_{d-2}=1}^{255} \mathsf{f}(\boldsymbol{l}|z_0, z_1, \ldots, z_{d-1}). \tag{51}$$

A plot of this PDF for $z = 0$ and $z = 1$ is given in Figures 41a and 41b. Such distributions typically correspond to a serial manipulation of the shares.

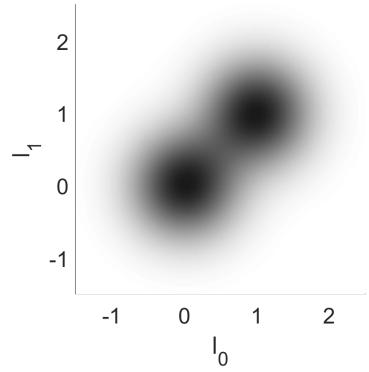
A parallel implementation can then be modeled by assuming that the manipulation of $d$ shares leads to noisy "sum of Hamming weight" leakage samples:

$$L_z^{\mathsf{par}} = \mathsf{HW}(z_0) + \mathsf{HW}(z_1) + \ldots + \mathsf{HW}(z_{d-1}) + r. \tag{52}$$

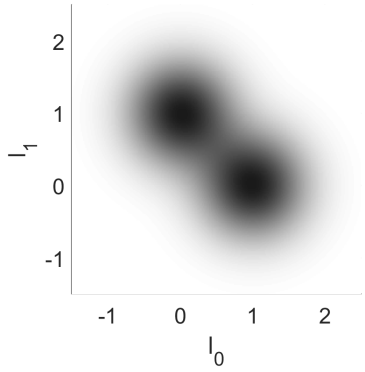
Such a model directly leads to the (univariate) distributions of Figures 41c and 41d. This figure is finally generalized to 3-share distributions in Figures 41e and 41f.

The interesting observation from these examples is that, for any $d$-share encoding, the statistical moment of order up to $d$-1 is independent of $z$. That is, in the 2-share cases, the mean vector of Figures 41a and 41b is $[0.5, 0.5]$ and the mean of 41c and 41d is 1. By contrast, the variances of these figures differ. And when moving to a 3-share case in Figures 41e and 41f, the variance becomes independent of $z$ and differences can only be spot by looking at the skewness.

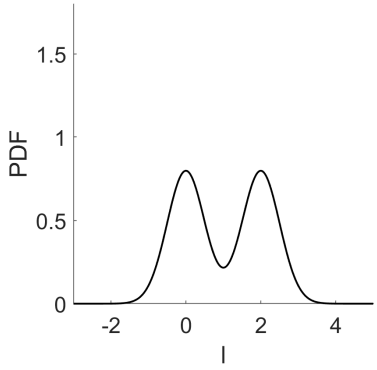
* Such an exhaustive computation is computationally hard. In practice, it is usually abandoned in favor of the convolution-based SASCA approach described in the previous section.



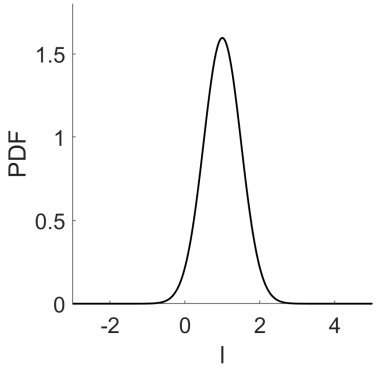

(a) Multivariate, 2 shares, $z = 0$.



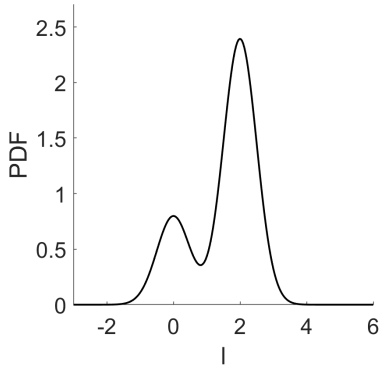

(b) Multivariate, 2 shares, $z = 1$.



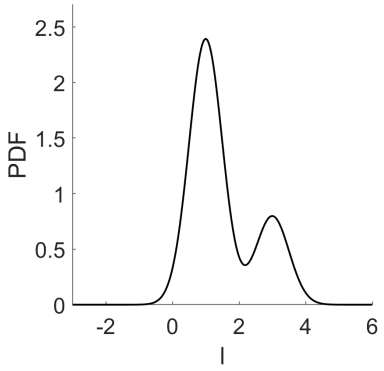

(c) Univariate, 2 shares $z = 0$.




(d) Univariate, 2 shares; $z = 1$.




(e) Univariate, 3 shares $z = 0$.




(f) Univariate, 3 shares; $z = 1$.

Figure 41: Leakage distributions of single-bit, 2- and 3-share encodings.

Informally, this means that an adversary willing to exploit the leakage of such masked implementations may have to estimate a statistical moment of order $d$, which implies raising the leakage samples to a power $d$. Combining this requirement with the definition of SNR in Section 2.2.1, Equation 3:

$$\mathsf{SNR} = \frac{\mathsf{Var}\big(\delta(Z_i)\big)}{\mathsf{Var}\big(R_i\big)}, \tag{53}$$

we can directly see that when SNR $< 1$, the estimation of a statistical moment of order $d$ will amplify the noise more than the signal. This intuitively implies that whenever adversaries are bound to a strategy based on moment estimation, the exponential security increase argued in Section 6.2.1 directly follows.

Such an intuition leads to the following definition of statistical security order:

**Definition 17** (Absolute statistical security Order)**.** *A masked implementation is said to have Absolute Statistical Security Order (ASSO) of order $o$ if the leakage distributions conditioned on any sensitive variable manipulated by the implementation only differ in statistical moments of orders strictly larger than $o$.*

It is important to observe that the ASSO is a heuristic requirement. First, a statistical distribution is not specified by its moments, making it hard to connect such a definition with formal security arguments. Second, and as already mentioned, an ASSO may not translate into concrete security: it only does if estimating a statistical moment is a meaningful adversarial strategy. Yet, and despite heuristic, we will see in Part III of the book that it is a useful notion to assess the independence assumption on which the security guarantees of masking rest.

We note finally that the word "absolute" refers to the fact that leakage dependencies in statistical moments of order lower than $o$ may not always degrade security (e.g., if these dependencies are small enough), which we will later capture with a notion of effective statistical security order. We also note that $o$ typically equals $d$-1 in the examples of Figure 41, but that may not always be the case.

### 6.2.3 Information theoretic analysis

We assess the impact of the masking countermeasure with an information theoretic analysis, using the leakage models given by Equations 50 and 52, which respectively capture a serial and a parallel manipulation of the shares. Contrary to the shuffling case, computing the MI for a known (here Hamming weight) leakage

model can be done relatively efficiently using Monte Carlo sampling or numerical integration (though computations naturally become more expensive as the estimated MI values decrease). We again use the SNR as a (normalized) metric reflecting the amount of additive noise. The results are reported in Figure 42.

The figure leads to three main intuitive observations that we detail next.

First, it confirms the exponential security amplification in the high noise (i.e., low SNR) regime claimed in the previous section. Note that the left parts of the plots, where the IT curves gradually become linear, exactly correspond to the cases where the aforementioned ASSO is meaningful. Under the perfect independence assumption that is guaranteed by our models, and since the plots are in log/log scale, the slope of the curves then correspond to the ASSO.

Second, and comparing Figure 42a with Figure 42b, we can see that the parallel manipulation of the shares leads to slightly less information than the serial one. This is expected given that parallel processing generally gives rise to less computation than serial processing, and therefore leads to less leakage.

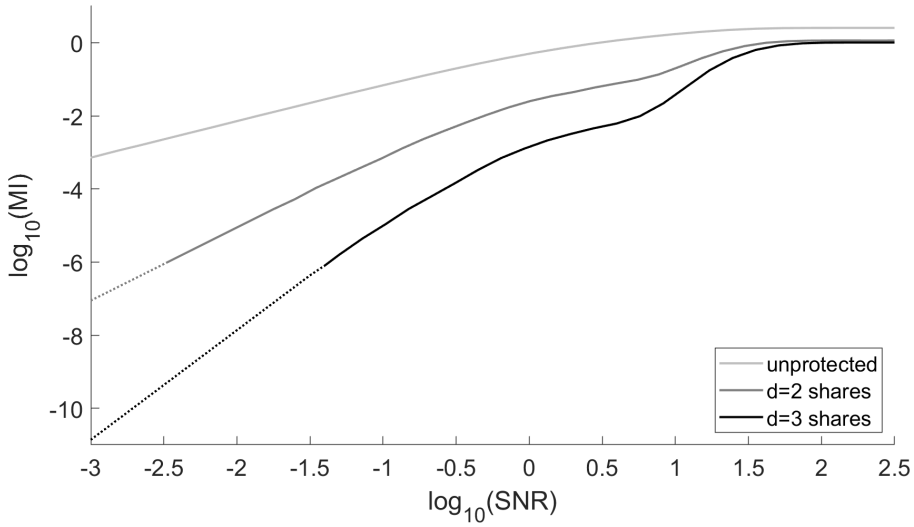
Finally, and as already hinted, the right parts of the plots show that without additive noise, Boolean masking is ineffective against Hamming weight leakage.

(*Exercise 2.11 is suggested in order to further illustrate the previous content.*)

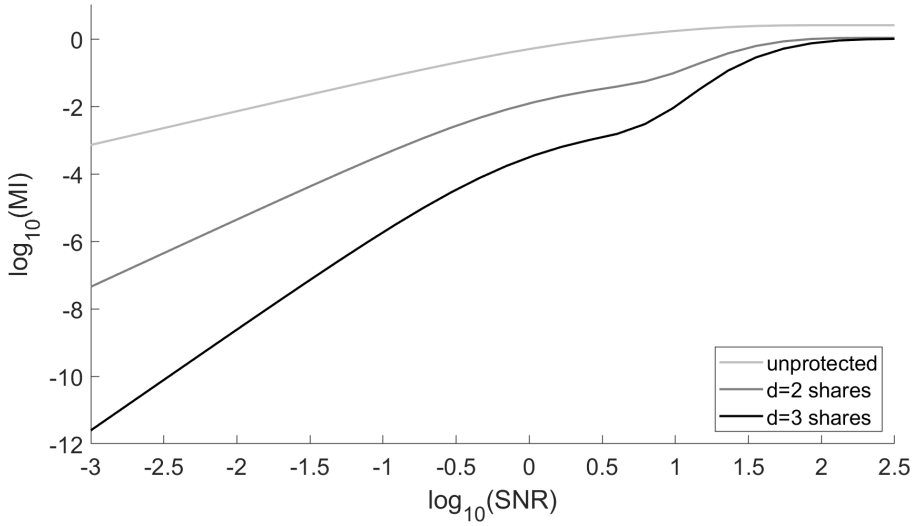
### 6.2.4 Discussion and related works

The results in this section show that the masking countermeasure can increase the security of a leaking implementation exponentially in a well-defined security parameter: the number of shares. However, this security is not unconditional. In contexts where Boolean encodings are used to share sensitive variables, which is popular for ciphers operating in binary fields like the AES, it is usually necessary that physical noise affects the measurements. So stepping back to this part of the book's introduction, Boolean masking cannot leverage any type of uncertainty. For example, a compressive leakage function like the Hamming weight one, despite not leaking its input in full (and therefore leaving uncertainty) is not enough.* We will see in the third part of the book that Boolean masking is also quite sensitive to an imperfectly satisfied independence assumption. Meanwhile, we can summarize the security guarantees of a Boolean encoding as follows:

* A similar observation holds for the generalized weighted sum of bits in Section 2.3.3.




(a) Serial shares' manipulation.




(b) Parallel shares' manipulation.

Figure 42: Information theoretic analysis of the masking countermeasure.

**Proposition 4** (Security of a Boolean encoding [DFS15])**.** *Let $Z$ be a sensitive variable and $\{Z_0, Z_1, \ldots, Z_{d-1}\}$ be a $d$-share Boolean encoding such that $Z_0, Z_1, \ldots,$*

$Z_{d-2} \leftarrow \{0,1\}^\eta$ *and* $Z_{d-1} = Z \oplus Z_0 \oplus Z_1 \oplus \ldots \oplus Z_{d-2}$*. Assume that each share gives rise to a leakage vector* $\boldsymbol{L}_i$ *satisfying the following assumptions:*

- *Uncertainty (noise). The mutual information per share* $\mathsf{MI}(Z_i; \boldsymbol{L}_i) \ll 1$*.*
- *Independence. The leakage vectors are generated as* $\boldsymbol{L}_i = \delta(Z_i) + \boldsymbol{R}_i$*, with* $\delta(.)$ *a deterministic leakage function and* $r_i$ *an additive noise;*

*Then, the mutual information on the sensitive variable* $Z$ *is bounded as:*

$$\mathsf{MI}(Z; \boldsymbol{L}) \leqslant \prod_{i=0}^{d-1} \mathsf{MI}(Z_i; \boldsymbol{L}_i). \tag{54}$$

A proof sketch of this proposition is already given in Section 6.2.1 and derives from the properties of the convolution product that is used to recombine the shares' information. By contrast, the study of its tightness is technical. Seed results with less tight parameters can be found in [PR13] and [DDF14]. The version we refer to in Proposition 4 was only conjectured in [DFS15]. A sequence of work then gradually refined its analysis [IUH22, MRS22], culminating with a generically tight bound in [BCG+23], which includes and additional factor $\big(2 \cdot \log(2)\big)^d$.*

Dropping this MI value into the bounds of Section 3.2.1 then directly turns Equation 54 into a bound on the number of traces needed for DPA to succeed.

The uncertainty (noise) condition is already discussed above: for the exponential security amplification of Boolean masking to take place, it is usually necessary that additive noise decreases the MI per share to reach the linear parts of the curves, as shown in Figure 42 for a Hamming weight leakage function.

The independence condition is quite intuitive too. If it is not respected and the leakage function recombines the shares, say as $L = \delta(Z_0 \oplus Z_1 \oplus \ldots \oplus Z_{d-1}) + R$, then the security of the encoding obviously collapses to the security of an unprotected implementation. Its formulation in proposition 4 may look too strong though. For example, it does not capture what happens for parallel implementations. This condition can however be relaxed thanks to the next lemma:

**Lemma 3** (Linear dimensionality reduction [BDF+17])**.** *Let* $\boldsymbol{L}$ *be the leakage vector of a* $d$*-share masked implementation. If this implementation guarantees ASSO* $t$*, then any implementation with leakage vector* $\boldsymbol{L}'$ *of which the samples are linear combinations of the samples of* $\boldsymbol{L}$ *in the reals also guarantees ASSO* $t$*.*

* By generically tight, we mean that there are some functions (e.g., bit leakages) for which the bound is tight, implying that it cannot be further improved without additional leakage assumptions. We will see next that it is not tight for the Hamming weight leakage function.

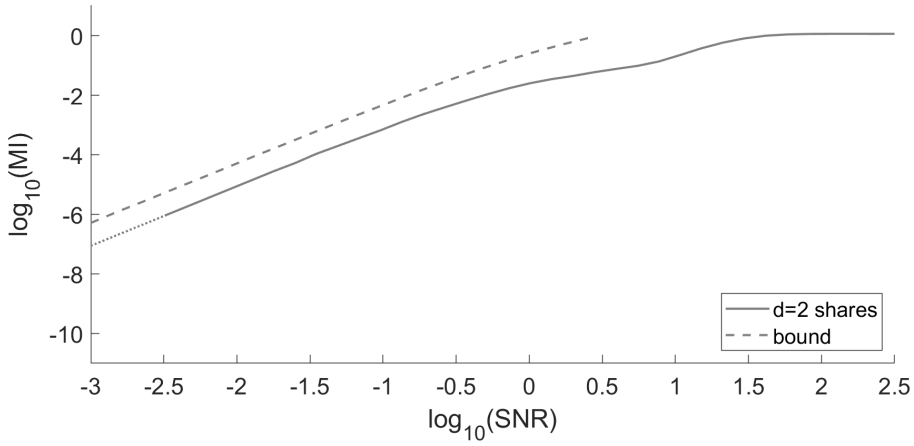
The proof is trivial. It requires the expectation of any polynomial of degree up to $o$ of the samples of $\boldsymbol{L}'$ to be independent of any sensitive variable manipulated by the implementation. This derives from Definition 17 which guarantees that any monomial of degree up to $o$ of the samples of $\boldsymbol{L}$ is independent of any sensitive variable manipulated by the implementation. As a typical example, a masked implementation manipulating its shares in parallel such that their leakage samples are only summed, as evaluated in Figure 42b, does not degrade security.

Figure 43 gives an illustration of Proposition 4 using the results of Figure 42 (unprotected curve) as an estimation of $\mathsf{MI}(Z_i; \boldsymbol{L}_i)$. One one can notice that the bound becomes more conservative as the number of shares increases.

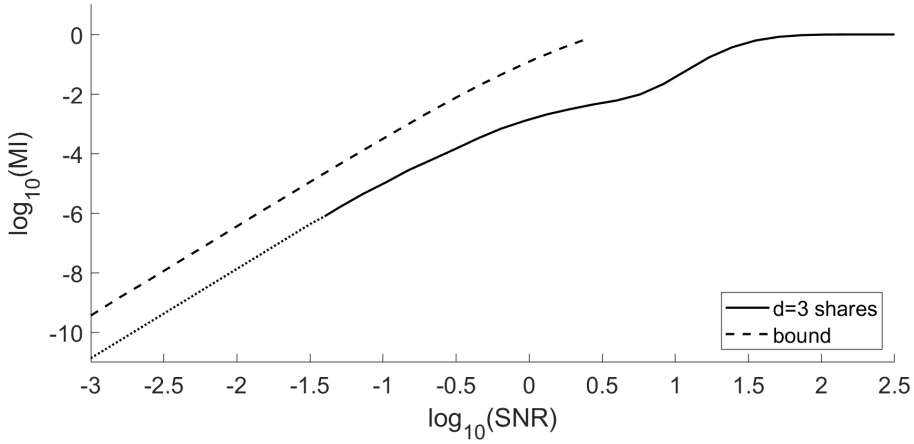
**Related works.** The notion of security order was first hinted in the 2-share case [Mes00]. The optimality of estimating a statistical moment for attacking masked implementation was first discussed in the 2-share case as well [PRB09]. The generalization of these findings and their relevance at higher security orders was then analyzed in [BGHR14]. Chari et al.'s seminal paper on masking already comes with a proof that it amplifies noise exponentially for a simplified (single-bit) power model inspired by Kocher et al.'s first DPA paper [CJRR99]. Various metrics (other than the MI) can be used to capture masking's uncertainty (noise) assumption. A comprehensive treatment can be found in [PGMP19]. There are also numerous alternatives to Boolean masking. Starting with the ones that are efficiently applicable to the AES, we can mention multiplicative masking [GT02], which nicely combines with arithmetic masking [GPQ11]. Following with the ones that are less efficiently applicable to the AES, we can mention affine masking [vW01, FMPR10], polynomial masking [PR11, GM11, GSF13], inner-product masking [DF11, BFG15, BFG$^+$17] and code-based masking [BCC$^+$14, WMCS20, WCG$^+$22]. These more complex encodings generally come with stronger security guarantees. Some of them are specific to certain (linear) leakage functions, like the security order amplification investigated in [WSY$^+$16, PGS$^+$17, CGC$^+$21]. Others are generic [FMM$^+$25]. Finally, it has been put forward that the algebraic compatibility between the field in which the shares are manipulated and the leakage function has a significant impact on masking in low noise (high SNR) settings. This motivated the investigation of prime-field masking [DFS16, MMMS23, FMM$^+$24].

## 6.3 Computing on masked data

The results of Section 6.2 show that masking data with (for example) Boolean encodings can offer strong guarantees against leakage. Yet, so far we only discussed the properties of a single encoding. For masking to be applicable to a block cipher like the AES, it remains to show that one can compute efficiently and securely




(a) Two serial shares.




(b) Three serial shares.

Figure 43: Information theoretic bound for the masking countermeasure.

on secret shared data. We will show in this section that this is achievable, despite raising significant challenges motivating the more formal investigations of Chapter 7 and Part III of the book. Given the conceptual complexity of analyzing masked computations directly, we first introduce a simple (abstract) model for this purpose. Its practical relevance will be discussed in Section 6.3.4.

In this model, denoted as the Threshold Probing Model (TPM), implementations are represented by a graph similar to the factor graphs of Section 2.4.1, where vertices are combinatorial gates and edges are wires carrying elements from a finite field $\mathbb{F}_{2^n}$. The adversary is allowed to read up to $t$ intermediate wires of a target implementation, leading to the next definition of security [ISW03]:

**Definition 18** (Threshold probing security)**.** *A circuit is $t$-probing secure iff every $t$-tuple of its intermediate variables is independent of any sensitive variable.*

By analogy with the ASSO, $t$ is referred to as the Probing Security Order (PSO).

### 6.3.1 Linear operations

Starting with the easiest case of a $d$-share encoding studied so far, it is easy to see that it directly leads to ($d$-1)-probing security. Namely, since:

$$z = z_0 \oplus z_1 \oplus \ldots \oplus z_{d-1}, \tag{55}$$

and $z_0, z_1, \ldots, z_{d-2}$ are picked up uniformly at random in $\mathbb{F}_{2^n}$, $z_i$ remains perfectly masked as long as a single share is not probed. It is also easy to see that this guarantee is maintained if one performs a linear operation on these shares. This is because the operation can then be applied independently on each share. In the AES case, say for example that $(e, f, g, h) = \mathsf{MixColumns}(a, b, c, d)$. A shared execution just requires to compute $(e_i, f_i, g_i, h_i) = \mathsf{MixColumns}(a_i, b_i, c_i, d_i)$, for $0 \leqslant i \leqslant d-1$. Since the shares remain isolated, this has no impact on the security in the TPM. Furthermore, the overheads of such linear operations only scale in $\mathcal{O}(d)$. That is, linearly in the number of shares, which is minimum.

### 6.3.2 Secure multiplications

Quite naturally, the situation starts to differ if non-linear operations like SubBytes must be executed. For this purpose, the by now standard option is to represent the multiplicative inverse of the AES S-box as a sequence of squarings and multiplications [RP10]. Since square operations are $\mathbb{F}_2$-linear, it is mainly the multiplication that requires special care. We will illustrate the challenges it raises with a 3-share example. For this purpose, let $a$ and $b$ be the sensitive values (bytes) we want to multiply, the $\otimes$ symbol denote a field multiplication and the $\boxtimes$ symbol

denote a shared multiplication. As a starting point, we can consider the following shared multiplication which is both secure and functionally correct:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \dot{\boxtimes} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} \xrightarrow{\text{prod.}} \begin{pmatrix} a_0 \otimes b_0 & a_0 \otimes b_1 & a_0 \otimes b_2 \\ a_1 \otimes b_0 & a_1 \otimes b_1 & a_1 \otimes b_2 \\ a_2 \otimes b_0 & a_2 \otimes b_1 & a_2 \otimes b_2 \end{pmatrix}.$$

Such a solution is however not satisfying. Indeed, it implies that the number of shares in the circuit is multiplied by $d$ (i.e., 3 in our example) each time a multiplication is performed. As a result, the cost of the implementation grows exponentially in the number of multiplications. So for the masking countermeasure to be cryptographically sound, we need a more efficient solution. Ignoring security for now, this can be achieved by compressing the matrix of $d^2$ partial products:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \ddot{\boxtimes} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} \xrightarrow{\text{prod.}} \begin{pmatrix} a_0 \otimes b_0 & a_0 \otimes b_1 & a_0 \otimes b_2 \\ a_1 \otimes b_0 & a_1 \otimes b_1 & a_1 \otimes b_2 \\ a_2 \otimes b_0 & a_2 \otimes b_1 & a_2 \otimes b_2 \end{pmatrix} \xrightarrow{\text{comp.}} \begin{pmatrix} \bigoplus_{i=0}^{2} a_0 \otimes b_i \\ \bigoplus_{i=0}^{2} a_1 \otimes b_i \\ \bigoplus_{i=0}^{2} a_2 \otimes b_i \end{pmatrix}.$$

Unfortunately, this time probing security is not maintained. By observing a single output share, the adversary has access to $a_i \otimes b$ which leaks information on the sensitive $b$.* So the solution Ishai et al. came up with is to re-randomize the partial products before compression, leading to the following gadget:

$$\begin{aligned} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \boxtimes \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} &\xrightarrow{\text{prod.}} \begin{pmatrix} a_0 \otimes b_0 & a_0 \otimes b_1 & a_0 \otimes b_2 \\ a_1 \otimes b_0 & a_1 \otimes b_1 & a_1 \otimes b_2 \\ a_2 \otimes b_0 & a_2 \otimes b_1 & a_2 \otimes b_2 \end{pmatrix}, \\ &\xrightarrow{\text{re-rand.}} \begin{pmatrix} a_0 \otimes b_0 & a_0 \otimes b_1 & a_0 \otimes b_2 \\ a_1 \otimes b_0 & a_1 \otimes b_1 & a_1 \otimes b_2 \\ a_2 \otimes b_0 & a_2 \otimes b_1 & a_2 \otimes b_2 \end{pmatrix} \oplus \begin{pmatrix} 0 & \varrho_{0,1} & \varrho_{0,2} \\ \varrho_{1,0} & 0 & \varrho_{1,2} \\ \varrho_{2,0} & \varrho_{2,1} & 0 \end{pmatrix}, \\ &\xrightarrow{\text{comp.}} \begin{pmatrix} \bigoplus_{i=0}^{2} (a_1 \otimes b_i \oplus \varrho_{0,i}) \\ \bigoplus_{i=0}^{2} (a_2 \otimes b_i \oplus \varrho_{1,i}) \\ \bigoplus_{i=0}^{2} (a_3 \otimes b_i \oplus \varrho_{2,i}) \end{pmatrix}, \end{aligned} \tag{56}$$

with $\varrho_{i,i=0}$ and $\varrho_{i,j} = \varrho_{j,i}$. This time, it can seen that each output share is masked by independent fresh randomness, which does not affect correctness since it adds a share of zero to the previous correct algorithm. We defer the formal security analysis of such a gadget to Chapter 7. Assuming it is ($d$-1)-probing secure for

* For example, in $\mathbb{F}_2$, $\otimes$ is the AND gate and observing $a_i \otimes b = 1$ allows concluding $b = 1$.

any $d$, it gives us first hints about the cryptographic relevance of the masking countermeasure, since it offers exponential security improvements at the cost of overheads scaling in $\mathcal{O}(d^2)$, both in size and in randomness. For completeness, we already give the generic specification of this gadget in Algorithm 2.

**Algorithm 2** Secure multiplication algorithm with $d \geqslant 2$ shares.

**Input:** shares $(a_i)_{0 \leqslant i \leqslant d-1}$ and $(b_i)_{0 \leqslant i \leqslant d-1}$, such that $\bigoplus_i a_i = a$ and $\bigoplus_j b_i = b$.
**Output:** shares $(c_i)_{0 \leqslant i \leqslant d-1}$, such that $\bigoplus_i c_i = a \otimes b$.

**for** $i = 0$ to $d-1$ **do**
  **for** $j = i+1$ to $d-1$ **do**
    $\varrho_{i,j} \xleftarrow{\$} \mathbb{F}_{2^n}$;
    $u_{i,j} \leftarrow (a_i \otimes b_j) \oplus \varrho_{i,j}$;
    $u_{j,i} \leftarrow (a_j \otimes b_i) \oplus \varrho_{i,j}$;
  **end for**
**end for**
**for** $i = 0$ to $d-1$ **do**
  $c_i \leftarrow a_i \otimes b_i \oplus \bigoplus_{j=0, j \neq i}^{d-1} u_{i,j}$;
**end for**

### 6.3.3 Composition challenges

Based on linear squaring gadgets performed share-wise and the multiplication gadget of Algorithm 2, an implementer's final goal would be to design a complete masked AES. In this section, we show that reaching this goal creates additional difficulties. At high level, this is because the composition of locally secure gadgets is not guaranteed to be secure. We illustrate this concern with the operation $x^3$ in $\mathbb{F}_{2^8}$, that is the beginning of the AES S-box and can be implemented by composing a square operation with a multiply one: $x^3 = x^2 \otimes x$. Considering again a 3-share example, one would expect that such a circuit is 3-probing secure. Concretely though, multiplying $x$ with $x^2$ implies computing the partial products:

$$\begin{pmatrix} x_0 \otimes (x_0)^2 & x_0 \otimes (x_1)^2 & x_0 \otimes (x_2)^2 \\ x_1 \otimes (x_0)^2 & x_1 \otimes (x_1)^2 & x_1 \otimes (x_2)^2 \\ x_2 \otimes (x_0)^2 & x_2 \otimes (x_1)^2 & x_2 \otimes (x_2)^2 \end{pmatrix}, \tag{57}$$

As in the previous section, there is a simple attack reducing the probing security order. Namely, an adversary observing $x_0 \otimes (x_1)^2$ and $x_2$ gains information about

$x$ with only two probes (rather than three as ideally expected). Intuitively, the origin of the problem lies in the fact that computing the cube implies multiplying dependent values. As for local multiplications, the solution to avoid such an attack is to re-randomize. A simple (though expensive) proposal for this purpose is to refresh one multiplication input by adding a share of zero as in Algorithm 2.

The attacks in this section and their (for now unproven) fixes motivate a formal treatment of security in the TPM and other (more realistic) models in Chapter 7. Meanwhile, we discuss the practical implications of a probing security flaw.

(*Exercises 2.12, 2.13 & 2.14 are suggested to illustrate the previous content.*)

### 6.3.4 Relevance of the probing model

Due to the abstract nature of the TPM, one can reasonably wonder whether a probing attack as described the previous sections has any concrete consequence on the security of an implementation against DPA. In order to discuss this question, we perform yet another information theoretic evaluation. This time, we compute the mutual information $\mathsf{MI}(X; \boldsymbol{L})$ between the sensitive variable $X$ in Section 6.3.3 and the Hamming weight leakage vector $\boldsymbol{L} = (L_{01}, L_3)$ of the two intermediate variables leading to a second-order probing attack, with:

$$L_{01} = \mathsf{HW}\big(X_0 \otimes (X_1)^2\big) + R_{0,1} \text{ and } L_3 = \mathsf{HW}(X_3) + R_3. \tag{58}$$

The result of this analysis is in Figure 44, where we performed the evaluation for a sensitive $X$ of different bit size $\eta$, leading to three main conclusions.

First, for $\eta = 4, 6, 8, 10$ and presumably larger values, reduced probing security translates into a reduced (ASSO and) security amplification in high noise (low SNR) regimes. This is reflected by the slopes -2 (vs. -3) of their IT curves.

Second, the link between the probing security order and ASSO does no hold for $\eta = 2$, which is due to a specific interaction between the square operation of the $X_0 \otimes (X_1)^2$ sample and the Hamming weight leakage function. It disappears if either the square is removed from the partial product or the Hamming weight function is modified even minimally (e.g., as a weighted sum of bits).

Third, it can be noticed that increasing the field size improves security against this attack vector in very low noise (high SNR) regimes. This is due to the fact that the informative events causing the statistical security order reduction relate to zero values that become rare as the field size grows. So it may be that for large fields and some SNR, targeting the encoding of $X$ with a third-order attack is a better strategy than targeting the composition flaw with a second-order attack.

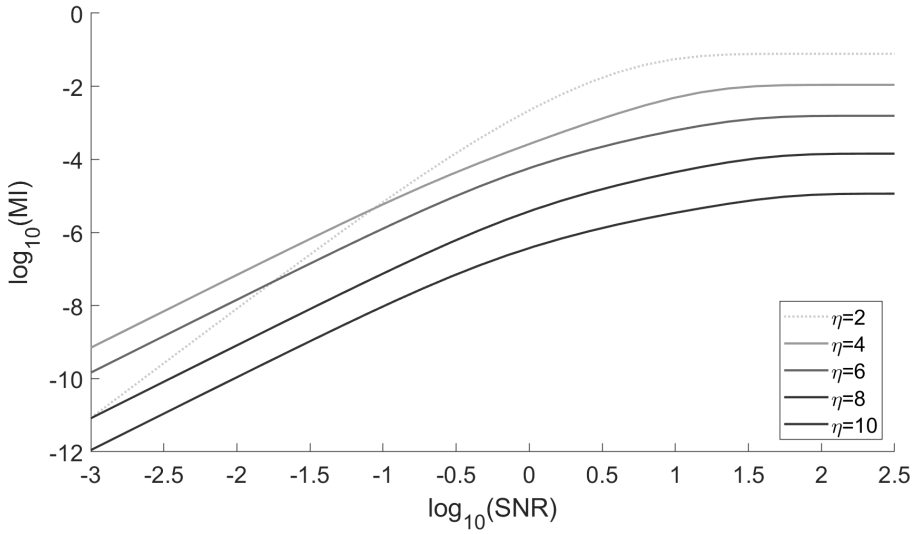



Figure 44: Information theoretic analysis of a probing security flaw.

Overall, this last analysis therefore highlights that probing security is not disconnected from concrete security. In particular in small fields, and besides leakage-specific exceptions, a flaw in the probing model generally translates into increased leakage (and reduced ASSO) that can be exploited by actual adversaries. Yet, the counter-example with $\eta = 2$ above also shows that it is not a strict/formal requirement: it is theoretically possible to construct implementations that are insecure in the TPM and secure in practice. As a side observation, the experiment in this section also suggests that security against certain attack vectors is possible without physical noise. This is the type of properties that the more complex encodings discussed in the related works part of Section 6.2.4 leverage.

### 6.3.5 Discussion and related work

The early sections of this chapter introduced the foundational principles that make masking a strong countermeasure. It is currently the only implementation-level solution that can be used to protect any cryptographic algorithm against the strong DPA attack vector, enabling exponential security amplification. This last section completed our claim of cryptographic soundness by emphasizing that it is possible to compute efficiently and securely over secret shared data. Yet, it also

showed that there is a significant gap between the tight understanding of leakage-resilient encodings, reflected by Proposition 4, and the complex challenges raised by non-linear computations and the composition thereof. As hinted above, this is already true in the abstract probing model. Chapter 7 and Part III of the book will show that these difficulties are further amplified when trying to argue quantitatively about the security level of masked implementations in more concrete leakage models, justifying the need of a more theoretical treatment.

**Related works.** Algorithm 2 is applicable to any field. Goudarzi and Rivain showed that, in the AES case, the bistlice representation given in Appendix A.1 leads to the faster implementations [GR17]. It is generally admitted that lightweight block ciphers leveraging an S-box with low multiplicative complexity and depth lead to more efficient masked implementations than the AES [GLSV14, BJK+16]. Given the simplicity and efficiency of masking linear operations, another approach towards improving DPA security is to leverage key-homomorphic re-keying schemes [MSGR10, DKM+15], which have been (crypt)analyzed in different models [DFH+16, DMMS21, HMM+23]. Masking requires large amounts of randomness, the generation of which is discussed in [CMM+24].

## 6.4 Combining countermeasures

# Chapter 7

# Provably secure masking

## 7.1 Threshold probing security

## 7.2 Random probing security

## 7.3 Noisy leakage security

## 7.4 Discussion and related works

# Exercises

## Paper-and-pencil

**Exercise 2.1** (*Concrete attacks' strength*). Consider a 128-bit implementation of the AES as for example used in Exercise 1.8 and illustrated in Figure 28. Quantify how much the SPA security against Key Recovery (KR) and State Distinguishing (SD) attacks differ. Assume 8-bit guesses and that only the AES plaintexts are given (i.e., not the ciphertexts) for the KR attack. Use the shortcut formulas from Section 3.2 for this purpose and discuss the tightness of the conclusions they lead to. Then try to extend this analysis to Message Distinguishing (MD) attacks.

**Exercise 2.2** (*CCAmL2 security*). Explain why it is reasonable to exclude the cases $N = N_{ch}$ in the post-challenge encryption queries and the cases $(N, c) = (N_{ch}, c_b)$ in the post-challenge decryption queries of Definition 9, Table 8.

**Exercise 2.3** (*SD attacks, revisited*). How much is the data complexity of the State Distinguishing (SD) attack discussed in Exercise 2.1 further improved if long messages are encrypted with the mode of Figure 29? (*Optional:* What is the impact of this observation for post-quantum encryption relying on the FO transform?)

**Exercise 2.4** (*Leakage-resilient MAC*). Describe a DPA attack that allows forging the MAC of a fresh message $\check{m}$ if the tag verification compares the (correct) recomputed $t$ with an adversarially-selected $t^*$ by checking that $\mathsf{HW}(t \oplus t^*) = 0$. Can this attack be avoided without leveraging an inverse-based verification?

**Exercise 2.5** (*Simulatable leakage*). Why is it necessary that the leakage simulator of Definition 16 is efficient for Proposition 3 to be practically-relevant?

**Exercise 2.6** (*Switching lemma*). Given that the AES is a PRP, why isn't the $\epsilon_{\mathsf{prp}}$ of Proposition 3 affected by a square root loss due to the PRP/PRF switching lemma?

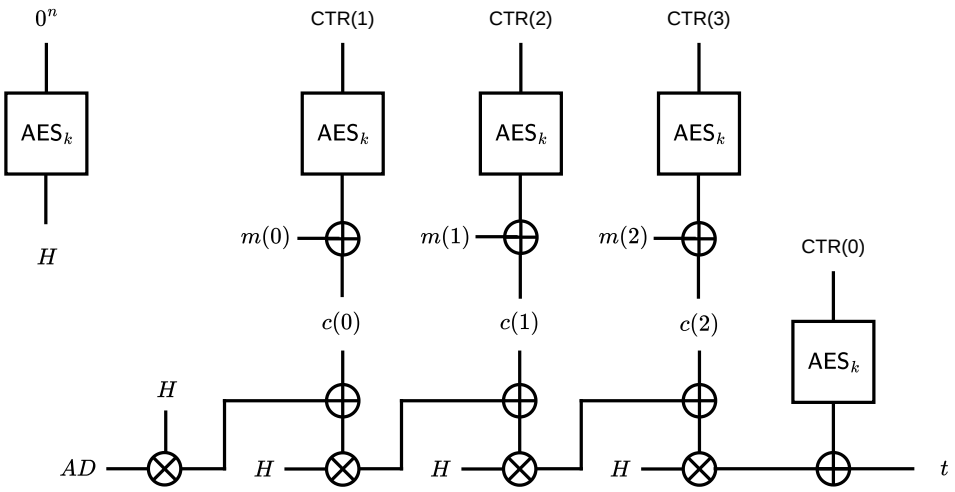



Figure 45: AES GCM mode of operation.

**Exercise 2.7** (*Leakage security of AES-GCM*). Perform a sensitivity analysis of the AES GCM mode of operation depicted in Figure 45 if it has to ensure CPAL1 or CIL1 security. That, is, determine the leakage requirements of all its blocks.

**Exercise 2.8** (*Confidentiality of the LR-PRF*). Assume that the LR-PRF of Figure 36 is directly used to encrypt the bits of a message $m$. How does its security against the Message Distinguishing (MD) attacks that unavoidably affect message's confidentiality with leakage compare to the one of TEDT (Figure 34)? Consider the same implementation as in Exercise 2.1 with Hamming weight leakages.

**Exercise 2.9** (*Shuffling vs. parallelism*). If a designer willing to prevent SPA against an AES implementation can choose between shuffling a serial (software) implementation and leveraging parallelism for a hardware implementation, what is the best choice from the security vs. performance tradeoff viewpoint?

**Exercise 2.10** (*Shuffling in the LRPM*). Quantify the security of the 4 solutions in Figure 39 using the Local Random Probing Model (LRPM) of Section 3.2.2.

**Exercise 2.11** (*Masking without noise*). Explain why increasing the number of shares does not improve security in the low noise (high SNR) parts of Figures 42a and 42b? What is the asymptotic value of the IT curves' plateau when $d$ increases? Is this weakness specific to the Hamming weight leakage function?

**Exercise 2.12** (*Masked AES S-box*). Propose a chain of multiplications and squarings to compute the multiplicative inverse of the AES S-box, minimizing the amount of multiplications. Assume that (i) multiplications are performed as suggested in Section 6.3.3; (ii) linear operations are performed share-wise; (iii) only the leakage of the operations' inputs and outputs can be exploited; (iv) the mutual information per share is $\Lambda$. Estimate the security of this implementation against an adversary exploiting the output $d$-tuple vs. all the $d$-tuples. How does the security evolve if one has to compose squarings to perform the linear operations?

**Exercise 2.13** (*Bitslice masking*) Compare the randomness requirements obtained for the solution of Exercise 2.12 with the ones of a bitsliced masked implementation taking advantage of the representation given in Appendix A, Table 13. Assume (sub-optimally) that all the multiplications are refreshed on both cases.

**Exercise 2.14** (*Master key sharing*) What is the probing security order of a 3-share implementation sharing its master key on-the-fly as $k_0 \leftarrow \$$, $k_1 \leftarrow \$$, $k_2 = k \oplus k_0 \oplus k_1$? How does it generalize to larger $d$ values? Can it be improved?

## Hands-on

Programming problems based on actual side-channel measurements of (software & hardware, unprotected, shuffled & masked) AES implementation, illustrating the challenges of ensuring the physical assumptions of this second part of the book, are available from: `https://github.com/simple-crypto/scale-two`. They are part of the SCALE-II training organized by the SIMPLE-Crypto association (see `https://www.simple-crypto.org/activities/scale/`).

# Part III

# Secure Implementations

# Introduction

# Chapter 8

# Ensuring DPA security

## 8.1 DPA security in hardware

## 8.2 DPA security in software

## 8.3 Discussion and related works

# Chapter 9

# Ensuring SPA security

## 9.1 SPA security in hardware

## 9.2 SPA security in software

## 9.3 Discussion and related works

# Chapter 10

# Backwards security evaluations

## 10.1 Testing DPA security

## 10.2 Testing SPA security

## 10.3 Discussion and related works

# Exercises

# Concluding remarks

## Scopes for further research

Starting with the first part of the book, each step in the side-channel analysis workflow of Figure 4 gives rise to interesting scopes for research. For the measurement and pre-processing step, an important one is the better understanding of attacks in the frequency (or other transformed) domain(s). While a couple of papers used such transforms for ad hoc reasons and show improvements in specific contexts, a systematic analysis allowing one to determine their pros and cons and the contexts in which they should be priviledged is still missing. For the detection and mapping step, and as recognized in Figure 3, it does not come with fundamental challenges if used as a preliminary in the (especially worst-case) analysis of unprotected implementations. Yet, the situation differs if detection is used in a conformance-style evaluation, in which case determining the threshold for which leakage is considered significant is challenging, in particular when analyzing highly multivariate traces. Developing statistically sound tests that lead to useful conclusions even when not detecting any leakage therefore still deserves further investigations. A similar observation holds for the modeling and extraction step: when analyzing "simple" leakage functions (e.g., linear combination of bits with Gaussian noise) as typically observed for unprotected implementations, Gaussian TA, PCA/LDA and linear regression generally provide excellent results in the worst-case evaluation setting. But how to best approach the efficiency of these information extraction tools without profiling, and how to generalize them to more complex leakage functions, remain important questions. Both are amplified when moving to secure implementations in Part III of the book. For the exploitation step, understanding (or getting rid of) the heuristics currently needed to perform a SASCA and a more formal treatment of adaptive attacks are pressing challenges. These difficulties are reflected in the theoretical background of Chapter 3, where the bounds and shortcuts for adaptive or analytical attacks re-

main missing or heuristic. As for the post-processing step, it is mostly the efficient enumeration of dependent secrets that would benefit from improvements.

## Link to hardware countermeasures

# Bibliography


[AARR02] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM Side-Channel(s). In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2002.

[ABB+20] 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 *SSR*, volume 12529 of *Lecture Notes in Computer Science*, pages 46–66. Springer, 2020.

[ABF13] Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. Leakage-Resilient Symmetric Encryption via Re-keying. In *CHES*, volume 8086 of *Lecture Notes in Computer Science*, pages 471–488. Springer, 2013.

[ABG+22] Melissa Azouaoui, Olivier Bronchain, Vincent Grosso, Kostas Papagiannopoulos, and François-Xavier Standaert. Bitslice Masking and Improved Shuffling: How and When to Mix Them in Software? *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(2):140–165, 2022.

[ABH+22] Melissa Azouaoui, Olivier Bronchain, Clément Hoffmann, Yulia Kuzovkova, Tobias Schneider, and François-Xavier Standaert. Systematic Study of Decryption and Re-encryption Leakage: The Case of Kyber. In *COSADE*, volume 13211 of *Lecture Notes in Computer Science*, pages 236–256. Springer, 2022.

[ADL17] Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting Authenticated Encryption Robustness with Minimal Modifications. In

*CRYPTO (3)*, volume 10403 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.

[ADP+20] 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 *INDOCRYPT*, volume 12578 of *Lecture Notes in Computer Science*, pages 205–227. Springer, 2020.

[APSQ06] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template Attacks in Principal Subspaces. In *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[APSV19] Melissa Azouaoui, Romain Poussier, François-Xavier Standaert, and Vincent Verneuil. Key Enumeration from the Adversarial Viewpoint. In *CARDIS*, volume 11833 of *Lecture Notes in Computer Science*, pages 252–267. Springer, 2019.

[ARR03] Dakshi Agrawal, Josyula R. Rao, and Pankaj Rohatgi. Multi-channel Attacks. In *CHES*, volume 2779 of *Lecture Notes in Computer Science*, pages 2–16. Springer, 2003.

[BBB+22] Francesco Berti, Shivam Bhasin, Jakub Breier, Xiaolu Hou, Romain Poussier, François-Xavier Standaert, and Balazs Udvarhelyi. A Finer-Grain Analysis of the Leakage (Non) Resilience of OCB. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):461–481, 2022.

[BBC+20] 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 *CRYPTO (1)*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020.

[BBGV16] Arthur Beckers, Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. Design and Implementation of a Waveform-Matching Based Triggering System. In *COSADE*, volume 9689 of *Lecture Notes in Computer Science*, pages 184–198. Springer, 2016.

[BCC+14] Julien Bringer, Claude Carlet, Hervé Chabanne, Sylvain Guilley, and Houssem Maghrebi. Orthogonal Direct Sum Masking - A Smartcard Friendly Computation Paradigm in a Code, with Builtin Protection against Side-Channel and Fault Attacks. In *WISTP*, volume 8501 of *Lecture Notes in Computer Science*, pages 40–56. Springer, 2014.

[BCG+23] Julien Béguinot, Wei Cheng, Sylvain Guilley, Yi Liu, Loïc Masure, Olivier Rioul, and François-Xavier Standaert. Removing the Field Size Loss from Duc et al.'s Conjectured Bound for Masked Encodings. In *COSADE*, volume 13979 of *Lecture Notes in Computer Science*, pages 86–104. Springer, 2023.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BDF+17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel Implementations of Masking Schemes and the Bounded Moment Leakage Model. In *EUROCRYPT (1)*, volume 10210 of *Lecture Notes in Computer Science*, pages 535–566, 2017.

[BDGN14] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. NICV: Normalized Inter-Class Variance for Detection of Side-Channel Leakage. In *International Symposium on Electromagnetic Compatibility*, pages 310–313, 2014.

[BDJR97] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *FOCS*, pages 394–403. IEEE Computer Society, 1997.

[BFG15] Josep Balasch, Sebastian Faust, and Benedikt Gierlichs. Inner Product Masking Revisited. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 486–510. Springer, 2015.

[BFG+17] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier Standaert. Consolidating inner product masking. In *ASIACRYPT (1)*, volume 10624 of *Lecture Notes in Computer Science*, pages 724–754. Springer, 2017.

[BGH+15] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Less is More - Dimensionality Reduction from a Theoretical Perspective. In *CHES*, volume 9293 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2015.

[BGHR14] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, and Olivier Rioul. Masks Will Fall Off - Higher-Order Optimal Distinguishers. In *ASIACRYPT (2)*, volume 8874 of *Lecture Notes in Computer Science*, pages 344–365. Springer, 2014.

[BGL08] Lejla Batina, Benedikt Gierlichs, and Kerstin Lemke-Rust. Comparative Evaluation of Rank Correlation Based DPA on an AES Prototype Chip. In *ISC*, volume 5222 of *Lecture Notes in Computer Science*, pages 341–354. Springer, 2008.

[BGL09] Lejla Batina, Benedikt Gierlichs, and Kerstin Lemke-Rust. Differential Cluster Analysis. In *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2009.

[BGP+19] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Strong Authenticity with Leakage Under Weak and Falsifiable Physical Assumptions. In *Inscrypt*, volume 12020 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2019.

[BGP+20] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT, a Leakage-Resist AEAD Mode for High Physical Security Applications. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):256–320, 2020.

[BGPS21] Francesco Berti, Chun Guo, Thomas Peters, and François-Xavier Standaert. Efficient Leakage-Resilient MACs Without Idealized Assumptions. In *ASIACRYPT (2)*, volume 13091 of *Lecture Notes in Computer Science*, pages 95–123. Springer, 2021.

[BGR95] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. In *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 1995.

[BHM+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage Certification Revisited: Bounding Model Errors in Side-Channel Security Evaluations. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 713–737. Springer, 2019.

[Bis07] Christopher M. Bishop. *Pattern Recognition and Machine Learning, 5th Edition*. Information science and statistics. Springer, 2007.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In *CRYPTO (2)*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[BKP08] Andrey Bogdanov, Ilya Kizhvatov, and Andrei Pyshkin. Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In *INDOCRYPT*, volume 5365 of *Lecture Notes in Computer Science*, pages 251–265. Springer, 2008.

[BKP+18] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext Integrity with Misuse and Leakage: Definition and Efficient Constructions with Symmetric Primitives. In *AsiaCCS*, pages 37–50. ACM, 2018.

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated Encryption in the Face of Protocol and Side Channel Leakage. In *ASIACRYPT (1)*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, 2017.

[BMP13] Joan Boyar, Philip Matthews, and René Peralta. Logic Minimization Techniques with Applications to Cryptology. *J. Cryptol.*, 26(2):280–312, 2013.

[BMPS21] Olivier Bronchain, Charles Momin, Thomas Peters, and François-Xavier Standaert. Improved Leakage-Resistant Authenticated Encryption based on Hardware AES Coprocessors. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):641–676, 2021.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.

[Bog07a] Andrey Bogdanov. Improved Side-Channel Collision Attacks on AES. In *Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2007.

[Bog07b] Andrey Bogdanov. Linear Slide Attacks on the KeeLoq Block Cipher. In *Inscrypt*, volume 4990 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2007.

[Bou26] Sana Boussam. Generic-compatible distinguishers for linear regression based attacks. In *CRYPTO (7)*, volume 16806 of *Lecture Notes in Computer Science*, pages 359–390. Springer, 2026.

[BPPS17] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On Leakage-Resilient Authenticated Encryption with Decryption Leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3):271–293, 2017.

[BPS19] Francesco Berti, Olivier Pereira, and François-Xavier Standaert. Reducing the Cost of Authenticity with Leakages: a \mathsf CIML2 -Secure \mathsf AE Scheme with One Call to a Strongly Protected Tweakable Block Cipher. In *AFRICACRYPT*, volume 11627 of *Lecture Notes in Computer Science*, pages 229–249. Springer, 2019.

[BRM+26] Julien Béguinot, Olivier Rioul, Loïc Masure, François-Xavier Standaert, Wei Cheng, and Sylvain Guilley. Scalable Information Theoretic Evaluation of the Rank Statistics in Side-Channel Attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2026:53–81, 2026.

[BS21] 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.

[BSH+14] Sonia Belaïd, Fabrizio De Santis, Johann Heyszl, Stefan Mangard, Marcel Medwed, Jörn-Marc Schmidt, François-Xavier Standaert,

and Stefan Tillich. Towards Fresh Re-Keying with Leakage-Resilient PRFs: Cipher Design Principles and Analysis. *J. Cryptogr. Eng.*, 4(3):157–171, 2014.

[BSL24] Francesco Berti, François-Xavier Standaert, and Itamar Levi. Authenticity in the Presence of Leakage using a Forkcipher. *IACR Commun. Cryptol.*, 1(4):13, 2024.

[BSS19] Olivier Bronchain, Tobias Schneider, and François-Xavier Standaert. Multi-Tuple Leakage Detection and the Dependent Signal Issue. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):318–345, 2019.

[BUS21] Davide Bellizia, Balazs Udvarhelyi, and François-Xavier Standaert. Towards a Better Understanding of Side-Channel Analysis Measurements Setups. In *CARDIS*, volume 13173 of *Lecture Notes in Computer Science*, pages 64–79. Springer, 2021.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In *CHES*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.

[CDP15] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Enhancing Dimensionality Reduction Methods for Side-Channel Attacks. In *CARDIS*, volume 9514 of *Lecture Notes in Computer Science*, pages 15–33. Springer, 2015.

[CDS23] Giovanni Camurati, Matteo Dell'Amico, and François-Xavier Standaert. MCRank: Monte Carlo Key Rank Estimation for Side-Channel Security Evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):277–300, 2023.

[CDSU23] Gaëtan Cassiers, Henri Devillez, François-Xavier Standaert, and Balazs Udvarhelyi. Efficient Regression-Based Linear Discriminant Analysis for Side-Channel Security Evaluations Towards Analytical Attacks against 32-bit Implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):270–293, 2023.

[CFGR12] Claude Carlet, Jean-Charles Faugère, Christopher Goyet, and Guénaël Renault. Analysis of the Algebraic Side-Channel Attack. *J. Cryptogr. Eng.*, 2(1):45–62, 2012.

[CGC+21] Wei Cheng, Sylvain Guilley, Claude Carlet, Sihem Mesnager, and Jean-Luc Danger. Optimizing Inner Product Masking Scheme by a Coding Theory Approach. *IEEE Trans. Inf. Forensics Secur.*, 16:220–235, 2021.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. An Efficient Method for Random Delay Generation in Embedded Software. In *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2009.

[CK10] Jean-Sébastien Coron and Ilya Kizhvatov. Analysis and improvement of the random delay countermeasure of CHES 2009. In *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 2010.

[CK13] Omar Choudary and Markus G. Kuhn. Efficient Template Attacks. In *CARDIS*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013.

[CK14] Omar Choudary and Markus G. Kuhn. Template Attacks on Different Devices. In *COSADE*, volume 8622 of *Lecture Notes in Computer Science*, pages 179–198. Springer, 2014.

[CK18] Marios O. Choudary and Markus G. Kuhn. Efficient, Portable Template Attacks. *IEEE Trans. Inf. Forensics Secur.*, 13(2):490–501, 2018.

[CLM20] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. Leakage Assessment Through Neural Estimation of the Mutual Information. In *ACNS Workshops*, volume 12418 of *Lecture Notes in Computer Science*, pages 144–162. Springer, 2020.

[CMM+24] Gaëtan Cassiers, Loïc Masure, Charles Momin, Thorben Moos, Amir Moradi, and François-Xavier Standaert. Randomness Generation for Secure Hardware Masking - Unrolled Trivium to the Rescue. *IACR Commun. Cryptol.*, 1(2):4, 2024.

[CP17] Marios O. Choudary and Pantelimon George Popescu. Back to Massey: Impressively Fast, Scalable and Tight Security Evaluation Tools. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 367–386. Springer, 2017.

[CPS16] Marios O. Choudary, Romain Poussier, and François-Xavier Standaert. Score-Based vs. Probability-Based Enumeration - A Cautionary Note. In *INDOCRYPT*, volume 10095 of *Lecture Notes in Computer Science*, pages 137–152, 2016.

[CRBO24] Aakash Chowdhury, Arnab Roy, Carlo Brunetta, and Elisabeth Oswald. Leakage Certification Made Simple. In *CRYPTO (6)*, volume 14925 of *Lecture Notes in Computer Science*, pages 427–460. Springer, 2024.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[CS19] 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.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition)*. Wiley, 2006.

[dCGRP19] 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.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. In *EURO-*

*CRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.

[DDF19] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. *J. Cryptol.*, 32(1):151–177, 2019.

[DDLM24] Nilanjan Datta, Avijit Dutta, Eik List, and Sougata Mandal. FEDT: Forkcipher-based Leakage-resilient Beyond-birthday-secure AE. *IACR Commun. Cryptol.*, 1(2):21, 2024.

[DEM+20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP v2.0. *IACR Trans. Symmetric Cryptol.*, 2020(S1):390–416, 2020.

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *J. Cryptol.*, 34(3):33, 2021.

[DF11] Stefan Dziembowski and Sebastian Faust. Leakage-Resilient Cryptography from the Inner-Product Extractor. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 702–721. Springer, 2011.

[DFH+16] Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards Sound Fresh Re-keying with Hard (Physical) Learning Problems. In *CRYPTO (2)*, volume 9815 of *Lecture Notes in Computer Science*, pages 272–301. Springer, 2016.

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.

[DFS16] Stefan Dziembowski, Sebastian Faust, and Maciej Skórski. Optimal Amplification of Noisy Leakages. In *TCC (A2)*, volume 9563 of *Lecture Notes in Computer Science*, pages 291–318. Springer, 2016.

[DHW+12] Benedikt Driessen, Ralf Hund, Carsten Willems, Christof Paar, and Thorsten Holz. Don't Trust Satellite Phones: A Security Analysis of Two Satphone Standards. In *IEEE Symposium on Security and Privacy*, pages 128–142. IEEE Computer Society, 2012.

[DJS19] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges Resist Leakage: The Case of Authenticated Encryption. In *ASIACRYPT (2)*, volume 11922 of *Lecture Notes in Computer Science*, pages 209–240. Springer, 2019.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On Cryptography with Auxiliary Input. In *STOC*, pages 621–630. ACM, 2009.

[DKM+15] Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security. In *CARDIS*, volume 9514 of *Lecture Notes in Computer Science*, pages 225–241. Springer, 2015.

[DM19] Christoph Dobraunig and Bart Mennink. Leakage Resilience of the Duplex Construction. In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2019.

[DM21] Christoph Dobraunig and Bart Mennink. Leakage Resilient Value Comparison with Application to Message Authentication. In *EUROCRYPT (2)*, volume 12697 of *Lecture Notes in Computer Science*, pages 377–407. Springer, 2021.

[DMMS21] Sébastien Duval, Pierrick Méaux, Charles Momin, and François-Xavier Standaert. Exploring Crypto-Physical Dark Matter and Learning with Physical Rounding Towards Secure and Efficient Fresh Re-Keying. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):373–401, 2021.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-Resilient Cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.

[DP10] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks.

In *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.

[DPRS11] 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.

[DR20] Joan Daemen and Vincent Rijmen. *The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition*. Information Security and Cryptography. Springer, 2020.

[DRS+12] François Durvaux, Mathieu Renauld, François-Xavier Standaert, Loïc van Oldeneel tot Oldenzeel, and Nicolas Veyrat-Charvillon. Efficient Removal of Random Delays from Embedded Software Implementations Using Hidden Markov Models. In *CARDIS*, volume 7771 of *Lecture Notes in Computer Science*, pages 123–140. Springer, 2012.

[DS09] Yevgeniy Dodis and John P. Steinberger. Message Authentication Codes from Unpredictable Block Ciphers. In *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 267–285. Springer, 2009.

[DS16] François Durvaux and François-Xavier Standaert. From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces. In *EUROCRYPT (1)*, volume 9665 of *Lecture Notes in Computer Science*, pages 240–262. Springer, 2016.

[DSE+12] Nicolas Debande, Youssef Souissi, M. Abdelaziz Elaabid, Sylvain Guilley, and Jean-Luc Danger. Wavelet Transform Based Pre-Processing for Side-Channel Analysis. In *MICRO Workshops*, pages 32–38. IEEE Computer Society, 2012.

[DSV+15] François Durvaux, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Jean-Baptiste Mairy, and Yves Deville. Efficient Selection of Time Samples for Higher-Order DPA with Projection Pursuits. In *COSADE*, volume 9064 of *Lecture Notes in Computer Science*, pages 34–50. Springer, 2015.

[DW17] Liron David and Avishai Wool. A Bounded-Space Near-Optimal Key Enumeration Algorithm for Multi-Subkey Side-Channel Attacks. In

*CT-RSA*, volume 10159 of *Lecture Notes in Computer Science*, pages 311–327. Springer, 2017.

[DW19a] Liron David and Avishai Wool. Fast Analytical Rank Estimation. In *COSADE*, volume 11421 of *Lecture Notes in Computer Science*, pages 168–190. Springer, 2019.

[DW19b] Liron David and Avishai Wool. Poly-Logarithmic Side Channel Rank Estimation via Exponential Sampling. In *CT-RSA*, volume 11405 of *Lecture Notes in Computer Science*, pages 330–349. Springer, 2019.

[DW22] Liron David and Avishai Wool. Rank Estimation with Bounded Error via Exponential Sampling. *J. Cryptogr. Eng.*, 12(2):151–168, 2022.

[DZD+17] A. Adam Ding, Liwei Zhang, François Durvaux, François-Xavier Standaert, and Yunsi Fei. Towards Sound and Optimal Leakage Detection Procedure. In *CARDIS*, volume 10728 of *Lecture Notes in Computer Science*, pages 105–122. Springer, 2017.

[EG12] M. Abdelaziz Elaabid and Sylvain Guilley. Portability of Templates. *J. Cryptogr. Eng.*, 2(1):63–74, 2012.

[ET93] Bradley Efron and Robert Tibshirani. *An Introduction to the Bootstrap*. Springer, 1993.

[FH15] Benjamin Fuller and Ariel Hamlin. Unifying Leakage Classes: Simulatable Leakage and Pseudoentropy. In *ICITS*, volume 9063 of *Lecture Notes in Computer Science*, pages 69–86. Springer, 2015.

[FLD12] Yunsi Fei, Qiasi Luo, and A. Adam Ding. A Statistical Model for DPA with Novel Algorithmic Confusion Analysis. In *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 233–250. Springer, 2012.

[FMM+24] Sebastian Faust, Loïc Masure, Elena Micheli, Maximilian Orlt, and François-Xavier Standaert. Connecting Leakage-Resilient Secret Sharing to Practice: Scaling Trends and Physical Dependencies of Prime Field Masking. In *EUROCRYPT (4)*, volume 14654 of *Lecture Notes in Computer Science*, pages 316–344. Springer, 2024.

[FMM+25] Sebastian Faust, Loïc Masure, Elena Michel, Hai Hoang Nguyen, Maximilian Orlt, and François-Xavier Standaert. IP Masking with Generic Security Guarantees Under Minimum Assumptions, and Applications. In *ASIACRYPT (2)*, volume 16246 of *Lecture Notes in Computer Science*, pages 544–575. Springer, 2025.

[FMPR10] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine Masking against Higher-Order Side Channel Analysis. In *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 262–280. Springer, 2010.

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical Leakage-Resilient Symmetric Cryptography. In *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, 2012.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual Information Analysis. In *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.

[GdKGM+08] Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE Classic. In *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 97–114. Springer, 2008.

[GG19] Cezary Glowacz and Vincent Grosso. Optimal Collision Side-Channel Attacks. In *CARDIS*, volume 11833 of *Lecture Notes in Computer Science*, pages 126–140. Springer, 2019.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions (Extended Abstract). In *FOCS*, pages 464–479. IEEE Computer Society, 1984.

[GGP+15] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and More Efficient Rank Estimation for Side-Channel Security Assessment. In *FSE*, volume 9054 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2015.

[GGSB20] 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.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A Testing Methodology for Side-Channel Resistance Validation, 2011.

[GLSV14] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. Ls-designs: Bitslice encryption for efficient masked software implementations. In *FSE*, volume 8540 of *Lecture Notes in Computer Science*, pages 18–37. Springer, 2014.

[GM11] Louis Goubin and Ange Martinelli. Protecting AES with Shamir's Secret Sharing Scheme. In *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2011.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[GO22] Si Gao and Elisabeth Oswald. A Novel Completeness Test for Leakage Models and Its Application to Side-Channel Attacks and Responsibly Engineered Simulators. In *EUROCRYPT (3)*, volume 13277 of *Lecture Notes in Computer Science*, pages 254–283. Springer, 2022.

[Gol11] Dieter Gollmann. *Computer Security (3. ed.)*. Wiley, 2011.

[GP99] Louis Goubin and Jacques Patarin. DES and Differential Power Analysis (The "Duplication" Method). In *CHES*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[GPPS19] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated Encryption with Nonce Misuse and Physical Leakage: Definitions, Separation Results and First Construction - (Extended Abstract). In *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.

[GPPS20] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards Low-Energy Leakage-Resistant Authenticated Encryption from the Duplex Sponge Construction. *IACR Trans. Symmetric Cryptol.*, 2020(1):6–42, 2020.

[GPQ11] Laurie Genelle, Emmanuel Prouff, and Michaël Quisquater. Thwarting Higher-Order Side Channel Analysis with Additive and Multiplicative Maskings. In *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 240–255. Springer, 2011.

[GPSG14] Vincent Grosso, Romain Poussier, François-Xavier Standaert, and Lubos Gaspar. Combining Leakage-Resilient PRFs and Shuffling - Towards Bounded Security for Small Embedded Devices. In *CARDIS*, volume 8968 of *Lecture Notes in Computer Science*, pages 122–136. Springer, 2014.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How Fast Can Higher-Order Masking Be in Software? In *EUROCRYPT (1)*, volume 10210 of *Lecture Notes in Computer Science*, pages 567–597, 2017.

[GRO18a] Joey Green, Arnab Roy, and Elisabeth Oswald. A Systematic Study of the Impact of Graphical Models on Inference-Based Attacks on AES. In *CARDIS*, volume 11389 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2018.

[Gro18b] Vincent Grosso. Scalable Key Rank Estimation (and Key Enumeration) Algorithm for Large Keys. In *CARDIS*, volume 11389 of *Lecture Notes in Computer Science*, pages 80–94. Springer, 2018.

[GS13] Benoît Gérard and François-Xavier Standaert. Unified and Optimized Linear Collision Attacks and their Application in a Non-Profiled Setting: Extended Version. *J. Cryptogr. Eng.*, 3(1):45–58, 2013.

[GS15] Vincent Grosso and François-Xavier Standaert. ASCA, SASCA and DPA with Enumeration: Which One Beats the Other and When? In *ASIACRYPT (2)*, volume 9453 of *Lecture Notes in Computer Science*, pages 291–312. Springer, 2015.

[GSF13] Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. Masking vs. multiparty computation: How large is the gap for aes? In *CHES*, volume 8086 of *Lecture Notes in Computer Science*, pages 400–416. Springer, 2013.

[GT02] Jovan Dj. Golic and Christophe Tymen. Multiplicative Masking and Power Analysis of AES. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2002.

[HGM+11] 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.

[HMM+23] Clément Hoffmann, Pierrick Méaux, Charles Momin, Yann Rotella, François-Xavier Standaert, and Balazs Udvarhelyi. Learning with Physical Rounding for Linear and Quadratic Leakage Functions. In *CRYPTO (3)*, volume 14083 of *Lecture Notes in Computer Science*, pages 410–439. Springer, 2023.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006.

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good Is Not Good Enough - Deriving Optimal Distinguishers from Communication Theory. In *CHES*, volume 8731 of *Lecture Notes in Computer Science*, pages 55–74. Springer, 2014.

[HSST23] Julius Hermelink, Silvan Streit, Emanuele Strieder, and Katharina Thieme. Adapting Belief Propagation to Counter Shuffling of NTTs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):60–88, 2023.

[HZ12] Annelie Heuser and Michael Zohner. Intelligent Machine Homicide - Breaking Cryptographic Devices Using Support Vector Machines. In *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 249–264. Springer, 2012.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[IUH22] Akira Ito, Rei Ueno, and Naofumi Homma. On the Success Rate of Side-Channel Attacks on Masked Implementations: Information-Theoretical Bounds and Their Practical Usage. In *CCS*, pages 1521–1535. ACM, 2022.

[JNP14] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In *ASIACRYPT (2)*, volume 8874 of *Lecture Notes in Computer Science*, pages 274–288. Springer, 2014.

[Jol86] Ian T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. 1986.

[Jou09] Antoine Joux. *Algorithmic Cryptanalysis*. Chapman and Hall/CRC, 2009.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[KPP20] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-Trace Attacks on Keccak. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):243–268, 2020.

[KR11] Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011.

[KR19] Yael Tauman Kalai and Leonid Reyzin. A Survey of Leakage-Resilient Cryptography. In *Providing Sound Foundations for Cryptography*, pages 727–794. ACM, 2019.

[KY00] Jonathan Katz and Moti Yung. Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 284–299. Springer, 2000.

[LBM14] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. Power Aalysis Attack: an Approach Based on Machine Learning. *Int. J. Appl. Cryptogr.*, 3(2):97–115, 2014.

[LCC+06] Thanh-Ha Le, Jessy Clédière, Cécile Canovas, Bruno Robisson, Christine Servière, and Jean-Louis Lacoume. A Proposition for Correlation Power Analysis Enhancement. In *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 174–186. Springer, 2006.

[Lis21] Eik List. TEDT2 - Highly Secure Leakage-Resilient TBC-Based Authenticated Encryption. In *LATINCRYPT*, volume 12912 of *Lecture Notes in Computer Science*, pages 275–295. Springer, 2021.

[LMV04] Hervé Ledig, Frédéric Muller, and Frédéric Valette. Enhancing Collision Attacks. In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2004.

[LP07] Kerstin Lemke-Rust and Christof Paar. Gaussian Mixture Models for Higher-Order Side-Channel Analysis. In *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 14–27. Springer, 2007.

[LPR13] Victor Lomné, Emmanuel Prouff, and Thomas Roche. Behind the Scene of Side-Channel Attacks. In *ASIACRYPT (1)*, volume 8269 of *Lecture Notes in Computer Science*, pages 506–525. Springer, 2013.

[LRW11] Moses D. Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable Block Ciphers. *J. Cryptol.*, 24(3):588–613, 2011.

[Mac03] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[Man04] Stefan Mangard. Hardware Countermeasures against DPA ? A Statistical Analysis of Their Effectiveness. In *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[Mas94] James L. Massey. Guessing and Entropy. In *IEEE International Symposium on Information Theory*, page 204, 1994.

[MCHS23] Loïc Masure, Gaëtan Cassiers, Julien M. Hendrickx, and François-Xavier Standaert. Information Bounds and Convergence Rates for Side-Channel Security Evaluators. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):522–569, 2023.

[MDB21] Macarena C. Martínez-Rodríguez, Ignacio M. Delgado-Lozano, and Billy Bob Brumley. SoK: Remote Power Analysis. In *ARES*, pages 7:1–7:12. ACM, 2021.

[Mes00] Thomas S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In *CHES*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

[MME10] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2010.

[MMMS23] Loïc Masure, Pierrick Méaux, Thorben Moos, and François-Xavier Standaert. Effective and Efficient Masking with Low Noise Using Small-Mersenne-Prime Ciphers. In *EUROCRYPT (4)*, volume 14007 of *Lecture Notes in Computer Science*, pages 596–627. Springer, 2023.

[MMO18] Daniel P. Martin, Luke Mather, and Elisabeth Oswald. Two Sides of the Same Coin: Counting and Enumerating Keys Post Side-Channel Attacks Revisited. In *CT-RSA*, volume 10808 of *Lecture Notes in Computer Science*, pages 394–412. Springer, 2018.

[MMOS16] Daniel P. Martin, Luke Mather, Elisabeth Oswald, and Martijn Stam. Characterisation and Estimation of the Key Rank Distribution in the Context of Side Channel Evaluations. In *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 548–572, 2016.

[MMR20] Thorben Moos, Amir Moradi, and Bastian Richter. Static Power Side-Channel Analysis - An Investigation of Measurement Factors. *IEEE Trans. Very Large Scale Integr. Syst.*, 28(2):376–389, 2020.

[MOBW13] Luke Mather, Elisabeth Oswald, Joe Bandenburg, and Marcin Wójcik. Does My Device Leak Information? An a priori Statistical Power Analysis of Leakage Detection Tests. In *ASIACRYPT (1)*, volume 8269 of *Lecture Notes in Computer Science*, pages 486–505. Springer, 2013.

[MOOS15] Daniel P. Martin, Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam. Counting Keys in Parallel After a Side Channel Attack. In *ASIACRYPT (2)*, volume 9453 of *Lecture Notes in Computer Science*, pages 313–337. Springer, 2015.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007.

[Mor14] Amir Moradi. Side-Channel Leakage through Static Power - Should We Care about in Practice? In *CHES*, volume 8731 of *Lecture Notes in Computer Science*, pages 562–579. Springer, 2014.

[MOS11] 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.

[MOSW15] Daniel P. Martin, Elisabeth Oswald, Martijn Stam, and Marcin Wójcik. A Leakage Resilient MAC. In *IMACC*, volume 9496 of *Lecture Notes in Computer Science*, pages 295–310. Springer, 2015.

[MOW14] Luke Mather, Elisabeth Oswald, and Carolyn Whitnall. Multi-target DPA Attacks: Pushing DPA Beyond the Limits of a Desktop Computer. In *ASIACRYPT (1)*, volume 8873 of *Lecture Notes in Computer Science*, pages 243–261. Springer, 2014.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking Cryptographic Implementations Using Deep Learning Techniques. In *SPACE*, volume 10076 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2016.

[MR04] Silvio Micali and Leonid Reyzin. Physically Observable Cryptography (Extended Abstract). In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[MRG+10] Olivier Meynard, Denis Réal, Sylvain Guilley, Florent Flament, Jean-Luc Danger, and Frédéric Valette. Characterization of the Electromagnetic Side-Channel in Frequency Domain. In *Inscrypt*, volume 6584 of *Lecture Notes in Computer Science*, pages 471–486. Springer, 2010.

[MRS22] Loïc Masure, Olivier Rioul, and François-Xavier Standaert. A Nearly Tight Proof of Duc et al.'s Conjectured Security Bound for Masked Implementations. In *CARDIS*, volume 13820 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2022.

[MRSS18] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. Leakage Detection with the x2-Test. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):209–237, 2018.

[MS16] Amir Moradi and François-Xavier Standaert. Moments-Correlating DPA. In *TISCCS*, pages 5–15. ACM, 2016.

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices. In *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.

[MSJ12] Marcel Medwed, François-Xavier Standaert, and Antoine Joux. Towards Super-Exponential Side-Channel Security with Efficient Leakage-Resilient PRFs. In *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 193–212. Springer, 2012.

[MSNF16] Marcel Medwed, François-Xavier Standaert, Ventzislav Nikov, and Martin Feldhofer. Unknown-Input Attacks in the Parallel Setting: Improving the Security of the CHES 2012 Leakage-Resilient PRF. In *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 602–623, 2016.

[MSV25] Charles Momin, François-Xavier Standaert, and Corentin Verhamme. Primitive-Level vs. Implementation-Level DPA Security: a Certified Case Study (Pleading for Standardized Leakage-Resilient Cryptography). *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(3):717–744, 2025.

[MWM21] Thorben Moos, Felix Wegener, and Amir Moradi. DL-LA: Deep Learning Leakage Assessment A Modern Roadmap for SCA Evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):552–598, 2021.

[Noe24] Mathieu Des Noes. Distribution of Signal to Noise Ratio and Application to Leakage Detection. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(2):384–402, 2024.

[OKPW10] Yossef Oren, Mario Kirschbaum, Thomas Popp, and Avishai Wool. Algebraic Side-Channel Analysis in the Presence of Errors. In *CHES*,

volume 6225 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2010.

[ORSW12] Yossef Oren, Mathieu Renauld, François-Xavier Standaert, and Avishai Wool. Algebraic Side-Channel Attacks Beyond the Hamming Weight Leakage Model. In *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 140–154. Springer, 2012.

[Pan03] Liam Paninski. Estimation of Entropy and Mutual Information. *Neural Comput.*, 15(6):1191–1253, 2003.

[Pea82] Judea Pearl. Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In *AAAI*, pages 133–136. AAAI Press, 1982.

[PGMP19] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. Unifying Leakage Models on a Rényi Day. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 683–712. Springer, 2019.

[PGS15] Romain Poussier, Vincent Grosso, and François-Xavier Standaert. Comparing Approaches to Rank Estimation for Side-Channel Security Evaluations. In *CARDIS*, volume 9514 of *Lecture Notes in Computer Science*, pages 125–142. Springer, 2015.

[PGS+17] Romain Poussier, Qian Guo, François-Xavier Standaert, Claude Carlet, and Sylvain Guilley. Connecting and Improving Direct Sum Masking and Inner Product Masking. In *CARDIS*, volume 10728 of *Lecture Notes in Computer Science*, pages 123–141. Springer, 2017.

[PHF09] Thomas Plos, Michael Hutter, and Martin Feldhofer. On Comparing Side-Channel Preprocessing Techniques for Attacking RFID Devices. In *WISA*, volume 5932 of *Lecture Notes in Computer Science*, pages 163–177. Springer, 2009.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):209–237, 2019.

[Pie09] Krzysztof Pietrzak. A Leakage-Resilient Mode of Operation. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.

[PP19] Peter Pessl and Robert Primas. More Practical Single-Trace Attacks on the Number Theoretic Transform. In *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 130–149. Springer, 2019.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 513–533. Springer, 2017.

[PR09] Emmanuel Prouff and Matthieu Rivain. Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In *ACNS*, volume 5536 of *Lecture Notes in Computer Science*, pages 499–518, 2009.

[PR11] Emmanuel Prouff and Thomas Roche. Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols. In *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2011.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against Side-Channel Attacks: A Formal Security Proof. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.

[Pro05] Emmanuel Prouff. DPA Attacks and S-Boxes. In *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 424–441. Springer, 2005.

[PS15] Santos Merino Del Pozo and François-Xavier Standaert. Blind Source Separation from Single Measurements Using Singular Spectrum Analysis. In *CHES*, volume 9293 of *Lecture Notes in Computer Science*, pages 42–59. Springer, 2015.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach. In *CHES*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016.

[PSKM15] Santos Merino Del Pozo, François-Xavier Standaert, Dina Kamel, and Amir Moradi. Side-channel attacks from static power: when should we care? In *DATE*, pages 145–150. ACM, 2015.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives. In *CCS*, pages 96–108. ACM, 2015.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[RBG+19] Debapriya Basu Roy, Shivam Bhasin, Sylvain Guilley, Annelie Heuser, Sikhar Patranabis, and Debdeep Mukhopadhyay. CC Meets FIPS: A Hybrid Test Methodology for First Order Side Channel Analysis. *IEEE Trans. Computers*, 68(3):347–361, 2019.

[RCN02] Jan Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits*. Prentice Hall. 2002.

[RGV12] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting Time Samples for Multivariate DPA Attacks. In *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 155–174. Springer, 2012.

[RGV17] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Fast Leakage Assessment. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 387–399. Springer, 2017.

[Riv08] Matthieu Rivain. On the Exact Success Rate of Side Channel Analysis in the Gaussian Model. In *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 165–183. Springer, 2008.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

[RPBC20] Prasanna Ravi, Romain Poussier, Shivam Bhasin, and Anupam Chattopadhyay. On Configurable SCA Countermeasures Against Single Trace Attacks for the NTT - A Performance Evaluation Study over Kyber and Dilithium on the ARM Cortex-M4. In *SPACE*, volume 12586 of *Lecture Notes in Computer Science*, pages 123–146. Springer, 2020.

[RPC22] Anca Radulescu, Pantelimon George Popescu, and Marios O. Choudary. GE vs GM: Efficient Side-Channel Security Evaluations on Full Cryptographic Keys. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):886–905, 2022.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):307–335, 2020.

[RS06] Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006.

[RSV09] Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009.

[RSV+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices. In *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2011.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In *CHES*, volume 5154

of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.

[SGV08] François-Xavier Standaert, Benedikt Gierlichs, and Ingrid Verbauwhede. Partition vs. Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices. In *ICISC*, volume 5461 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 2008.

[SKS09] François-Xavier Standaert, François Koeune, and Werner Schindler. How to Compare Profiled Side-Channel Attacks? In *ACNS*, volume 5536 of *Lecture Notes in Computer Science*, pages 485–498, 2009.

[SLFP04] Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. A Collision Attack on AES: Combining Side-Channel and Differential Attack. In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 163–175. Springer, 2004.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side-Channel Cryptanalysis. In *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.

[SM16] Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - Extended Version. *J. Cryptogr. Eng.*, 6(2):85–99, 2016.

[SMSG16] Tobias Schneider, Amir Moradi, François-Xavier Standaert, and Tim Güneysu. Bridging the Gap: Advanced Tools for Side-Channel Leakage Estimation Beyond Gaussian Templates and Histograms. In *SAC*, volume 10532 of *Lecture Notes in Computer Science*, pages 58–78. Springer, 2016.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[SÖP04] François-Xavier Standaert, Siddika Berna Örs, and Bart Preneel. Power Analysis of an FPGA: Implementation of Rijndael: Is Pipelin-

ing a DPA Countermeasure? In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 30–44. Springer, 2004.

[SPAQ06] François-Xavier Standaert, Eric Peeters, Cédric Archambeau, and Jean-Jacques Quisquater. Towards Security Limits in Side-Channel Attacks. In *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 2006.

[SPRQ06] François-Xavier Standaert, Eric Peeters, Gaël Rouvroy, and Jean-Jacques Quisquater. An Overview of Power Analysis Attacks Against Field Programmable Gate Arrays. *Proc. IEEE*, 94(2):383–394, 2006.

[SPS+22] Yaobin Shen, Thomas Peters, François-Xavier Standaert, Gaëtan Cassiers, and Corentin Verhamme. Triplex: an Efficient and One-Pass Leakage-Resistant Mode of Operation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):135–162, 2022.

[SPS24] Yaobin Shen, Thomas Peters, and François-Xavier Standaert. Multiplex: TBC-Based Authenticated Encryption with Sponge-Like Rate. *IACR Trans. Symmetric Cryptol.*, 2024(2):1–34, 2024.

[SPY+10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage Resilient Cryptography in Practice. In *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 99–134. Springer, 2010.

[SPY13] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-Resilient Symmetric Cryptography under Empirically Verifiable Assumptions. In *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013.

[SS23] Yaobin Shen and François-Xavier Standaert. Optimally Secure Tweakable Block Ciphers with a Large Tweak from n-bit Block Ciphers. *IACR Trans. Symmetric Cryptol.*, 2023(2):47–68, 2023.

[Sta18] François-Xavier Standaert. How (Not) to Use Welch's T-Test in Side-Channel Security Evaluations. In *CARDIS*, volume 11389 of *Lecture Notes in Computer Science*, pages 65–79. Springer, 2018.

[Ste46] Stanley Smith Stevens. On the Theory of Scales of Measurement. *Science*, 103(2684):677–680, 1946.

[TOT+14] Sébastien Tiran, Sébastien Ordas, Yannick Teglia, Michel Agoyan, and Philippe Maurine. A Model of the Leakage in the Frequency Domain and its Application to CPA and DPA. *J. Cryptogr. Eng.*, 4(3):197–212, 2014.

[USS+20] Florian Unterstein, Marc Schink, Thomas Schamberger, Lars Tebelmann, Manuel Ilg, and Johann Heyszl. Retrofitting Leakage Resilient Authenticated Encryption to Microcontrollers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):365–388, 2020.

[UvWBS20] Balazs Udvarhelyi, Antoine van Wassenhove, Olivier Bronchain, and François-Xavier Standaert. On the Security of Off-the-Shelf Microcontrollers: Hardware Is Not Enough. In *CARDIS*, volume 12609 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 2020.

[UXT+22] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. Curse of Re-encryption: A Generic Power/EM Analysis on Post-Quantum KEMs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):296–322, 2022.

[VGRS12] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In *Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2012.

[VGS13] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security Evaluations Beyond Computing Power. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2013.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft Analytical Side-Channel Attacks. In *ASIACRYPT (1)*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note. In *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.

[VS09] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Mutual Information Analysis: How, When and Why? In *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 429–443. Springer, 2009.

[VS10] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Adaptive Chosen-Message Side-Channel Attacks. In *ACNS*, volume 6123 of *Lecture Notes in Computer Science*, pages 186–199, 2010.

[vW01] Manfred von Willich. A Technique with an Information-Theoretic Basis for Protecting Secret Data from Differential Power Attacks. In *IMACC*, volume 2260 of *Lecture Notes in Computer Science*, pages 44–62. Springer, 2001.

[Wag12] Mathias Wagner. 700+ Attacks Published on Smart Cards: The Need for a Systematic Counter Strategy. In *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 33–38. Springer, 2012.

[WCG+22] Qianmei Wu, Wei Cheng, Sylvain Guilley, Fan Zhang, and Wei Fu. On Efficient and Secure Code-based Masking: A Pragmatic Evaluation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(3):192–222, 2022.

[WMCS20] Weijia Wang, Pierrick Méaux, Gaëtan Cassiers, and François-Xavier Standaert. Efficient and Private Computations with Code-Based Masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):128–171, 2020.

[WO15] Carolyn Whitnall and Elisabeth Oswald. Robust Profiling for DPA-Style Attacks. In *CHES*, volume 9293 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2015.

[WO19] 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 *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 256–284. Springer, 2019.

[WOM11] Carolyn Whitnall, Elisabeth Oswald, and Luke Mather. An Exploration of the Kolmogorov-Smirnov Test as a Competitor to Mutual Information Analysis. In *CARDIS*, volume 7079 of *Lecture Notes in Computer Science*, pages 234–251. Springer, 2011.

[WOS14] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. The Myth of Generic DPA...and the Magic of Learning. In *CT-RSA*, volume 8366 of *Lecture Notes in Computer Science*, pages 183–205. Springer, 2014.

[WSY+16] Weijia Wang, François-Xavier Standaert, Yu Yu, Sihang Pu, Junrong Liu, Zheng Guo, and Dawu Gu. Inner Product Masking for Bitslice Ciphers and Security Order Amplification for Linear Leakages. In *CARDIS*, volume 10146 of *Lecture Notes in Computer Science*, pages 174–191. Springer, 2016.

[WYS+18] Weijia Wang, Yu Yu, François-Xavier Standaert, Junrong Liu, Zheng Guo, and Dawu Gu. Ridge-Based DPA: Improvement of Differential Power Analysis For Nanoscale Chips. *IEEE Trans. Inf. Forensics Secur.*, 13(5):1301–1316, 2018.

[XPR+22] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David F. Oswald, Wang Yao, and Zhiming Zheng. Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems With Chosen Ciphertexts: The Case Study of Kyber. *IEEE Trans. Computers*, 71(9):2163–2176, 2022.

[YEM14] Xin Ye, Thomas Eisenbarth, and William Martin. Bounded, yet Sufficient? How to Determine Whether Limited Side Channel Information Enables Key Recovery. In *CARDIS*, volume 8968 of *Lecture Notes in Computer Science*, pages 215–232. Springer, 2014.

[YMO22] Rebecca Young, Luke Mather, and Elisabeth Oswald. Comparing Key Rank Estimation Methods. In *CARDIS*, volume 13820 of *Lecture Notes in Computer Science*, pages 188–204. Springer, 2022.

[You12] William Henry Young. On the Multiplication of Successions of Fourier Constants. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 87(596):331–339, October 1912.

[YS13] Yu Yu and François-Xavier Standaert. Practical Leakage-Resilient Pseudorandom Objects with Minimum Public Randomness. In *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2013.

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In *CCS*, pages 141–151. ACM, 2010.

[YTM+20] Yuan Yao, Michael Tunstall, Elke De Mulder, Anton Kochepasov, and Patrick Schaumont. Augmenting Leakage Detection Using Bootstrapping. In *COSADE*, volume 12244 of *Lecture Notes in Computer Science*, pages 104–119. Springer, 2020.

[ZDF20] Ziyue Zhang, A. Adam Ding, and Yunsi Fei. A Fast and Accurate Guessing Entropy Estimation Algorithm for Full-key Recovery. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):26–48, 2020.

# Appendix A

# The Advanced Encryption Standard

## A.1 Specifications

In October 2000, the US National Institute of Standards and Technology selected the block cipher Rijndael as the Advanced Encryption Standard (AES). In this appendix, we recall the general structure of the AES and its main internal operations. Our descriptions follow the more detailed ones of [DR20]. By default, we consider the AES with 128-bit plaintexts $x$ and 128-bit key $k$. The AES is an iterated block cipher: it consists in the repeated application of a round transformation, denoted as Round, which is itself a sequence of four operations on the 128-bit state: SubBytes, ShiftRows, MixColumns and AddRoundKey:

Round($x$, $ExpandedKey[i]$)
{
    SubBytes($x$);
    ShiftRows($x$);
    MixColumns($x$);
    AddRoundKey($x$,$ExpandedKey[i]$);
}

The MixColumns operation is removed from the final round and denoted as FinalRound, leading to the following high-level algorithmic description:

AES($x$, $k$)
{
    KeyExpansion($k$, $ExpandedKey$);

AddRoundKey($x$,$ExpandedKey[0]$);
for $\varrho = 1; \varrho < N_r; \varrho++$) Round($x$, $ExpandedKey[\varrho]$);
FinalRound($x$, $ExpandedKey[N_r]$);
}

The 128-bit version of the AES iterates ten rounds to produce a ciphertext. Its three first rounds are illustrated in Figure 46 where the $x_i$'s denote the sixteen plaintext bytes and $k_i^\varrho$ denotes the byte $i$ of the expanded key at round $\varrho$. We do not detail the AES key expansion algorithm which will not be needed for our discussions and next briefly describe the round's inner operations.

SubBytes is the only non-linear part of the cipher. It consists in the parallel application of sixteen S-boxes to the bytes of the cipher state. The S-boxes implement a multiplicative inverse in $\mathbb{F}_{2^8}$ followed by an affine transform. It benefits from various representations. Of particular interest in the context of this book are the representations as sequences of multiplication and squaring operations, which are handy for masked implementations [RP10]. Table 13, also gives the bitslice representation of Boyar et al. [BMP13] which can serve for the same purpose.

ShiftRows is a byte transposition that cyclically shifts the four rows of the state over offsets ranging from zero to three in the 128-bit version of the AES.

MixColumns is a bricklayer permutation operating on the 4-byte columns of the state. It considers these columns as a polynomial over $\mathbb{F}_{2^8}$ which is multiplied modulo $x^4 + 1$ with a constant polynomial. This multiplication can be written as the following matrix multiplication, which is illustrated in Figure 47:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \text{'02'} & \text{'03'} & \text{'01'} & \text{'01'} \\ \text{'01'} & \text{'02'} & \text{'03'} & \text{'01'} \\ \text{'01'} & \text{'01'} & \text{'02'} & \text{'03'} \\ \text{'03'} & \text{'01'} & \text{'01'} & \text{'02'} \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}, \tag{59}$$

where the values '01', '02' and '03' are the hexadecimal representation of the polynomials $1$, $x$ and $1 \oplus x$ over $\mathbb{F}_{2^8} \equiv \mathbb{F}_2/p(x)$, with $p(x) = x^8 \oplus x^4 \oplus x^3 \oplus x \oplus 1$. The multiplication by '02', denoted as xtimes, has a simple Boolean expression which allows efficient bitslice implementations [GR17]. Denoting the bits of a byte $b$ as $\big(b(7), b(6), b(5), b(5), b(3), b(2), b(1), b(0)\big)$, xtimes$(b)$ is worth:

$$\big(b(6), b(5), b(4), b(3) \oplus b(7), b(2) \oplus b(7), b(1), b(0) \oplus b(7), b(7)\big). \tag{60}$$

AddRoundKey is a 128-bit bitwise XOR between the state and the round key.

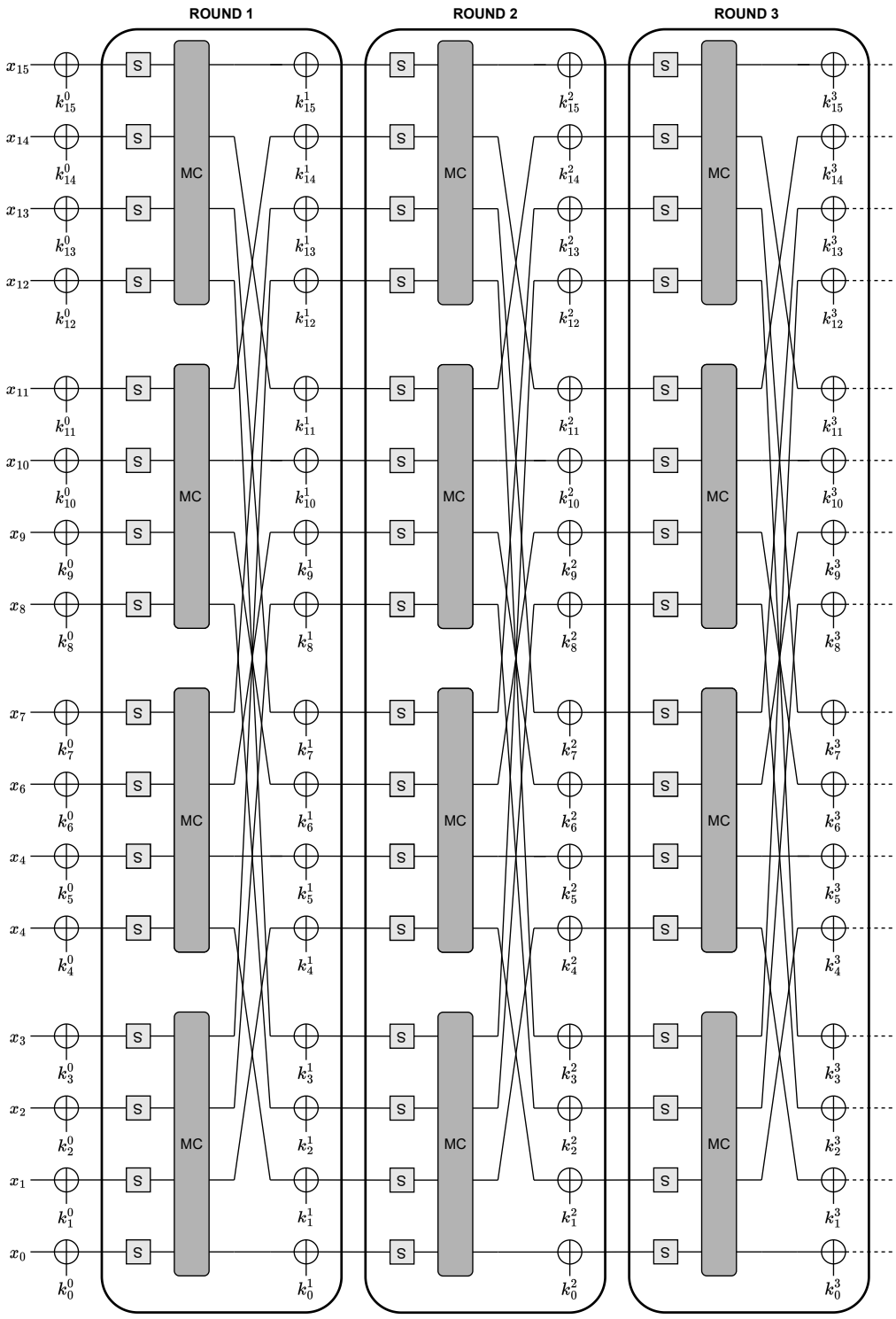



Figure 46: Three first rounds of the AES Rijndael (wires are bytes).

| | | | |
|---|---|---|---|
| $b14 = a3 \oplus a5$ | $b1 = c0 \oplus a7$ | $b15 = c1 \oplus a5$ | $b17 = b10 \oplus b11$ |
| $b13 = a0 \oplus a6$ | $b4 = b1 \oplus a3$ | $b20 = c1 \oplus a1$ | $b19 = b10 \oplus b8$ |
| $b12 = b13 \oplus b14$ | $b2 = b1 \oplus a0$ | $b6 = b15 \oplus a7$ | $b16 = c0 \oplus b11$ |
| $b9 = a0 \oplus a3$ | $b5 = b1 \oplus a6$ | $b10 = b15 \oplus c0$ | $b21 = b13 \oplus b16$ |
| $b8 = a0 \oplus a5$ | $c1 = a4 \oplus b12$ | $b11 = b20 \oplus b9$ | $b18 = a0 \oplus b16$ |
| $c0 = a1 \oplus a2$ | $b3 = b5 \oplus b8$ | $b7 = a7 \oplus b11$ | |

(a) Top linear layer.

| | | | |
|---|---|---|---|
| $c2 = b12 \otimes b15$ | $c23 = c19 \oplus b21$ | $c34 = c23 \oplus c33$ | $c2 = c33 \otimes a7$ |
| $c3 = b3 \otimes b6$ | $c15 = b8 \otimes b10$ | $c35 = c27 \oplus c33$ | $c3 = c43 \otimes b16$ |
| $c5 = b4 \otimes a7$ | $c26 = c21 \otimes c23$ | $c42 = c29 \oplus c33$ | $c4 = c40 \otimes b1$ |
| $c7 = b13 \otimes b16$ | $c16 = c15 \oplus c12$ | $d14 = c29 \otimes b2$ | $d6 = c42 \otimes b11$ |
| $c8 = b5 \otimes b1$ | $c18 = c6 \oplus c16$ | $c36 = c24 \otimes c35$ | $d7 = c45 \otimes b17$ |
| $c10 = b2 \otimes b7$ | $c20 = c11 \oplus c16$ | $c37 = c36 \oplus c34$ | $d8 = c41 \otimes b10$ |
| $c12 = b9 \otimes b11$ | $c24 = c20 \oplus b18$ | $c38 = c27 \oplus c36$ | $d9 = c44 \otimes b12$ |
| $c13 = b14 \otimes b17$ | $c30 = c23 \oplus c24$ | $c39 = c29 \otimes c38$ | $d10 = c37 \otimes b3$ |
| $c4 = c3 \oplus c2$ | $c22 = c18 \oplus b19$ | $d5 = c29 \otimes b7$ | $d11 = c33 \otimes b4$ |
| $c6 = c5 \oplus c2$ | $c25 = c21 \oplus c22$ | $c44 = c33 \oplus c37$ | $d12 = c43 \otimes b13$ |
| $c9 = c8 \oplus c7$ | $c27 = c24 \oplus c26$ | $c40 = c25 \oplus c39$ | $d13 = c40 \otimes b5$ |
| $c11 = c10 \oplus c7$ | $c31 = c22 \oplus c26$ | $c41 = c40 \oplus c37$ | $d15 = c42 \otimes b9$ |
| $c14 = c13 \oplus c12$ | $c28 = c25 \otimes c27$ | $c43 = c29 \oplus c40$ | $d16 = c45 \otimes b14$ |
| $c17 = c4 \oplus c14$ | $c32 = c31 \otimes c30$ | $c45 = c42 \oplus c41$ | $d17 = c41 \otimes b8$ |
| $c19 = c9 \oplus c14$ | $c29 = c28 \oplus c22$ | $d0 = c44 \otimes b15$ | |
| $c21 = c17 \oplus b20$ | $c33 = c32 \oplus c24$ | $d1 = c37 \otimes b6$ | |

(b) Non-linear layer.

| | | | |
|---|---|---|---|
| $c46 = d15 \oplus d16$ | $c49 = d9 \oplus d10$ | $c61 = d14 \oplus c57$ | $c48 = d5 \oplus d13$ |
| $c55 = d16 \oplus d17$ | $c63 = c49 \oplus c58$ | $c65 = c61 \oplus c62$ | $c56 = d12 \oplus c48$ |
| $c52 = d7 \oplus d8$ | $c66 = d1 \oplus c63$ | $e0 = c59 \oplus c63$ | $e3 = c53 \oplus c66$ |
| $c54 = d6 \oplus d7$ | $c62 = c52 \oplus c58$ | $c51 = d2 \oplus d5$ | $e1 = \overline{c64 \oplus s3}$ |
| $c58 = d4 \oplus c46$ | $c53 = d0 \oplus d3$ | $e4 = c51 \oplus c66$ | $e6 = \overline{c56 \oplus c62}$ |
| $c59 = d3 \oplus c54$ | $c50 = d2 \oplus d12$ | $e5 = c47 \oplus c65$ | $e7 = \overline{c48 \oplus c60}$ |
| $c64 = d4 \oplus c59$ | $c57 = c50 \oplus c53$ | $c67 = c64 \oplus c65$ | |
| $c47 = d10 \oplus d11$ | $c60 = c46 \oplus c57$ | $e2 = \overline{c55 \oplus c67}$ | |

(c) Bottom linear layer.

Table 13: Bitslice AES S-box from [BMP13].

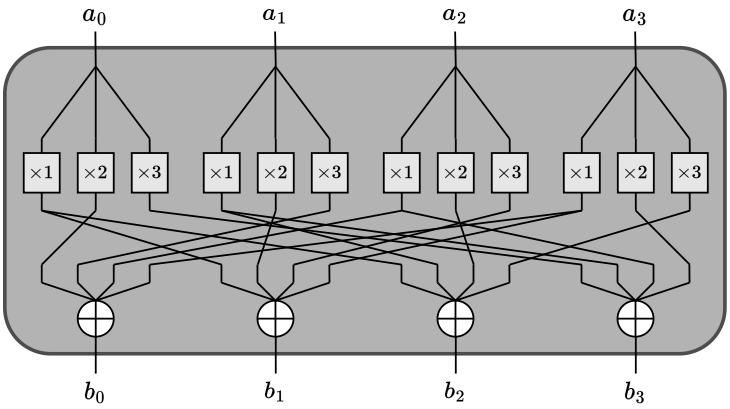



Figure 47: The AES MixColumns operation (wires are bytes).

## A.2 Tweakable versions

Some of our expositions (e.g., when discussing leakage-resistant modes of operation in Part II) benefit from the possibility to leverage a tweakable block cipher. For simplicity, we denote as TAES1 (resp., TAES2) a TBC using the same rounds as the AES (but a few more of them) with 128-bit (resp., 256-bit) tweaks and a tweakey expansion algorithm following the framework of [JNP14]. We do not detail the tweakey expansion algorithm of the TAES1 and TAES2 since, as the key expansion off the AES, they will not be needed for our discussions.