Side-Channel Analysis and Leakage-Resistance

François-Xavier Standaert

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

Version 1.2, September 2024

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 verified 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 the security proofs. 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 varying 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 readers with the theoretical and practical background which will allow them integrating these best solutions in various contexts and navigating through the side-channel analysis and leakage-resistance literature.

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 countermeasures 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 **10**

Introduction **11**

1 Exemplary attack **13**

2 Evaluation tools **18**
2.1 Measurement and pre-processing 20
2.2 Detection and mapping 22
2.3 Modeling and extraction 32
2.4 Information exploitation 42
2.5 Post-processing 47

3 Theoretical background **53**
3.1 Side-channel analysis framework 53
3.2 Bounds and shortcuts 58
3.3 Computing the metrics 61
3.4 Discussion and related works 66

Exercises **71**

II Leakage-Resistance 76

Introduction 77

4 Security definitions 78
4.1 Completing the attacks' taxonomy 78
4.2 Integrity with leakage 78
4.3 Confidentiality with leakage 78
4.4 Authenticated encryption 78

5 Mode-level and primitive-level countermeasures 79
5.1 Leakage-resistant MACs 79
5.2 Leakage-resistant encryption 79
5.3 Leakage-resistant AE 79
5.4 Primitive-level optimizations 79

6 Implementation-level countermeasures 80
6.1 Masking basics 80
6.2 Shuffling 80
6.3 Combining countermeasures 80
6.4 Masking theory 80

Exercises 81

III Secure Implementations 82

Introduction 83

7 Hardware designs 84
7.1 Ensuring DPA security 84
7.2 Ensuring SPA security 84

8 Software designs 85
8.1 Ensuring DPA security 85
8.2 Ensuring SPA security 85

9 Backwards security evaluations **86**
9.1 Testing DPA security 86
9.2 Testing SPA security 86
9.3 Discussion and related works 86

Exercises **87**

Concluding remarks **88**

Bibliography **89**

Appendix A The Advanced Encryption Standard **108**
A.1 Specifications 108
A.2 Tweakable versions 112

List of acronyms

AES	Advanced Encryption Standard
ASIC	Application-Specific Integrated Circuit
BP	Belief Propagation
CPA	Correlation Power Analysis
DPA	Differential Power Analysis
DUT	Device Under Test
FPGA	Field Programmable Gate Array
HW	Hamming weight
KeS	Key Sensitive (variable)
KIS	Key and Input Sensitive (variable)
LRPM	Local Random Probing Model
LDA	Linear Discriminant Analysis
MI	Mutual Information
PCA	Principal Component Analysis
PCB	Printed Circuit Board
PDF	Probability Density Function
POI	Point-of-Interest (in a leakage trace)
repSPA	Simple Power Analysis with repetition
RPM	Random Probing Model
SASCA	Soft Analytical Side-Channel Attack
SNR	Signal-to-Noise Ratio
SPA	Simple Power Analysis
TA	Template Attack
TAES1	Tweakable AES with 128-bit tweak
TAES2	Tweakable AES with 256-bit tweak
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 16×2^8 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 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. A leakage trace $\boldsymbol{l}$ obtained by measuring 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 contains $\tau = 300$ time samples denoted as l_j. Starting with the first S-box for illustration, an adversary can define a plaintext vector $\boldsymbol{x}$ of size q,

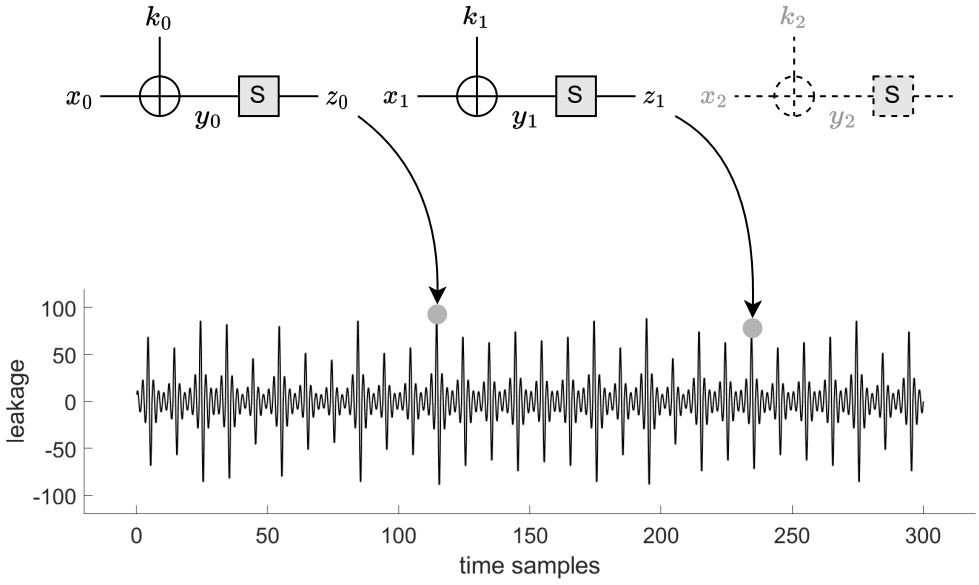


Figure 1: Leakage trace $\boldsymbol{l}$ for the AES' first two S-boxes ($\tau = 300$ samples).

the corresponding plaintext byte vector $\boldsymbol{x_0}$ and, for every key byte candidate $k_0^* \in [0:255]$, compute (q-element) model vectors $\boldsymbol{m}_j^{z_0}(\boldsymbol{x_0}, k_0^*) = \mathsf{HW}\big(\mathsf{S}(\boldsymbol{x_0} \oplus k_0^*)\big)$ with HW the Hamming weight function. By collecting the leakage traces corresponding to the q plaintexts of $\boldsymbol{x}$, she can then build a leakage matrix $\bar{\boldsymbol{l}}(\boldsymbol{x}, k)$ of size $q \times \tau$. The model vectors $\boldsymbol{m}_j^{z_0}(\boldsymbol{x_0}, k_0^*)$ can be viewed as q samples of a univariate random variable $M_j^{z_0}(X_0, k_0^*) = \mathsf{HW}\big(\mathsf{S}(X_0 \oplus k_0^*)\big)$, where plaintext bytes are random and key byte candidates are fixed, and the leakage matrix $\bar{\boldsymbol{l}}(\boldsymbol{x}, k)$ can be viewed as q samples of a multivariate random variable $\boldsymbol{L}(X, k)$. Eventually, the adversary selects her key byte candidate based on Pearson's correlation:

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

which is computed for each time sample independently. It can be estimated from q samples thanks the sample mean $\hat{\mathsf{E}}_q(.)$ and sample variance $\hat{\mathsf{Var}}_q(.)$ as:

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

Let us consider the following plaintext byte vector for illustration:

$$\boldsymbol{x_0} = [\,0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\,],$$

with $q = 11$. It leads to the following models (identical for all time samples):

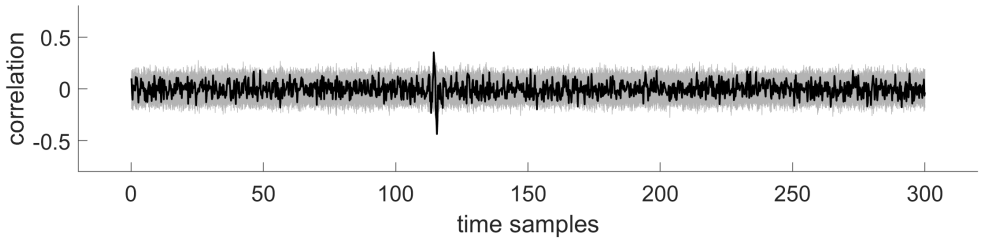
$$\begin{aligned}
\boldsymbol{m}_j^{z_0}(\boldsymbol{x_0}, k_0^* = 0) &= [\,4,\ 5,\ 6,\ 6,\ 5,\ 5,\ 6,\ 4,\ 2,\ 1,\ 5\,],\\
\boldsymbol{m}_j^{z_0}(\boldsymbol{x_0}, k_0^* = 1) &= [\,5,\ 4,\ 6,\ 6,\ 5,\ 5,\ 4,\ 6,\ 1,\ 2,\ 4\,],\\
\boldsymbol{m}_j^{z_0}(\boldsymbol{x_0}, k_0^* = 2) &= [\,6,\ 6,\ 4,\ 5,\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

where the values for $k_0^* = 0$ are computed as $\mathsf{HW}\big(\mathsf{S}(0 \oplus 0)\big) = 4$, $\mathsf{HW}\big(\mathsf{S}(1 \oplus 0)\big) = 5$, … The attack then uses Pearson's correlation to compare the 256 vectors $\boldsymbol{m}_j^{z_0}(\boldsymbol{x_0}, k_0^*)$ and the leakage measured for the target key byte $\bar{\boldsymbol{l}}(\boldsymbol{x}, k)$. Say that the correct key byte $k_0 = 2$ and assume that the leakage of an intermediate variable is indeed correlated with its Hamming weight, which is an acceptable approximation for unprotected CMOS devices [BCO04]. Then, for the time sample 115 where the output of the first S-box is manipulated, we should have:

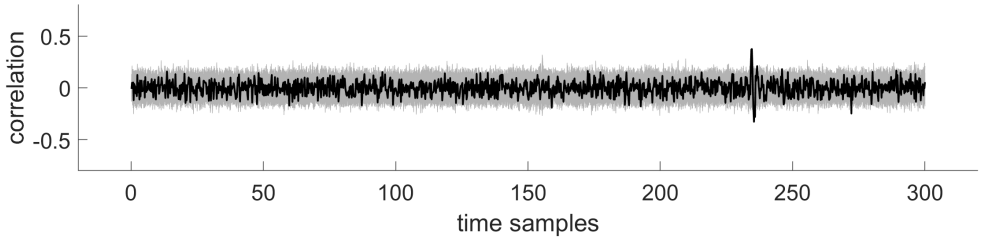
$$\boldsymbol{l}_{115}(\boldsymbol{x}, k) \approx [\,6,\ 6,\ 4,\ 5,\ 6,\ 4,\ 5,\ 5,\ 5,\ 4,\ 2\,] + \boldsymbol{r}_{115},$$

where $\boldsymbol{r}_{115}$ is an 11-dimension unknown random sampling, capturing the fact that actual measurements are generally affected by some physical noise. As a result, the highest correlation should be observed for the correct key byte (once well-enough estimated). The argmax of Equation 1 is additionally taken over the time samples if the adversary does not know a priori the mapping between the time samples of the leakage traces and the operations performed by the target implementation. Figure 2 shows the result of such a correlation attack after the observation of $q = 256$ traces (rather than $q = 11$ in the example above).

Note that, as mentioned in the preface, the traces we use are simulated in order to simplify their interpretation. 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. 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.



(a) First S-box.



(b) Second S-box.

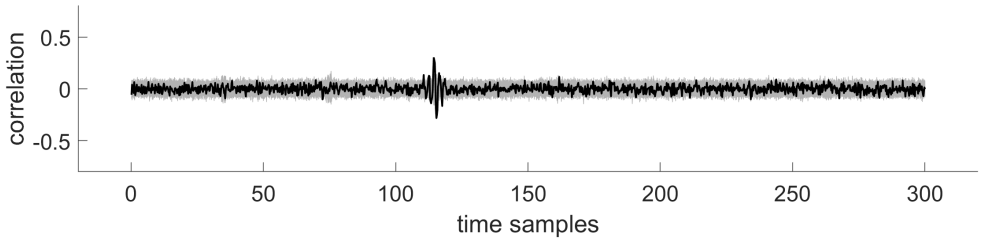
Figure 2: Correlation between the Hamming weights of the AES S-boxes' outputs ($\forall$ key byte candidates) and leakage traces for the correct key byte ($q = 256$).

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 is explained by the presence of additive noise in our simulated traces. For uniformly distributed plaintexts, the same traces can finally be used to recover the other k_i's (with $i \in [1 : 15]$) by adapting the models $\boldsymbol{m}_j^{z_i}(\boldsymbol{x_i}, k_i^*)$ and selecting key byte candidates as in Equation 1.

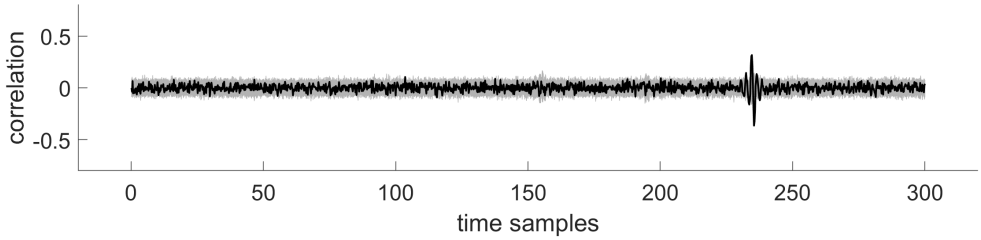
(*Exercise 1.1 is suggested in order to further illustrate the previous content.*)

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)?



(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 ($q = 1024$).

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.

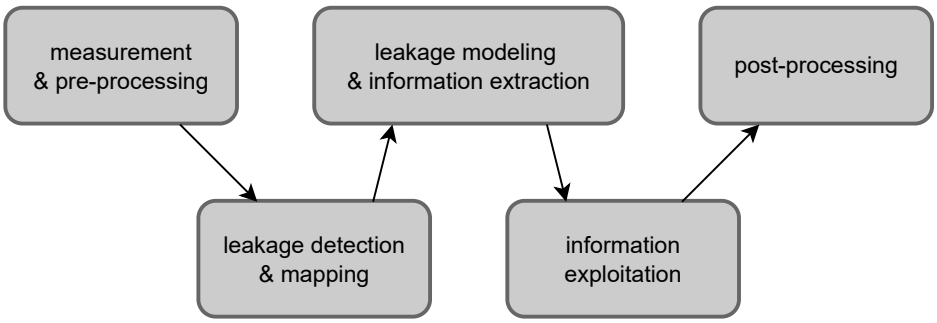


Figure 4: Side-channel analysis workflow.

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, since 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)**.** *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 the knowledge of the algorithms is now accepted as the standard model for mathematical cryptanalysis. By contrast, 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 (possibly unknown) optimal attack paths in this case. 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.

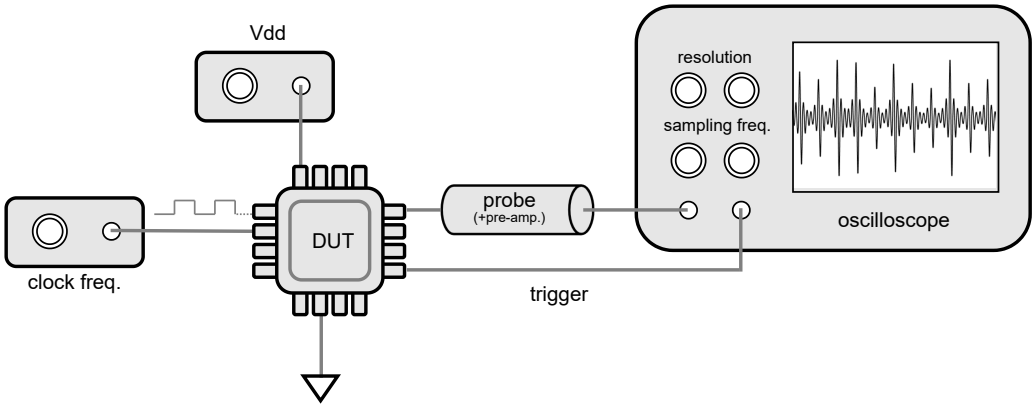


Figure 5: Side-channel measurement setup.

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 these guidelines are for standard setups exploiting the dynamic power consump-

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

Related works. 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, the latter comes 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 estimation 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 δ_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 θ samples, respectively. In our AES case study where the target intermediate value is 8-bit wide, the q traces collected in order to estimate the signal variance are split in

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

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:

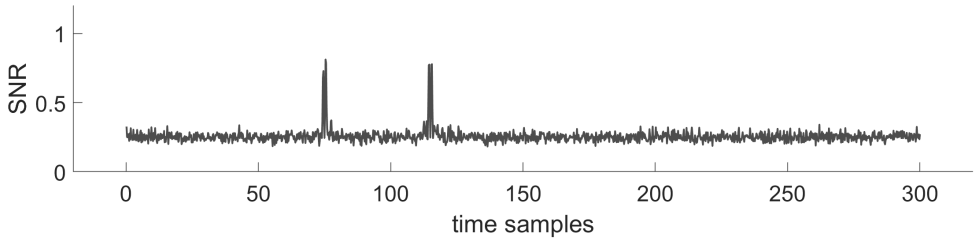
Definition 3 (Sensitive variables)**.** *We say that an intermediate value in a cryptographic implementation is Key and Input Sensitive (KIS) if it can be expressed as a function of a group operation (e.g., a XOR) between a part of the algorithm's secret key and a part of its public input (e.g., plaintext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

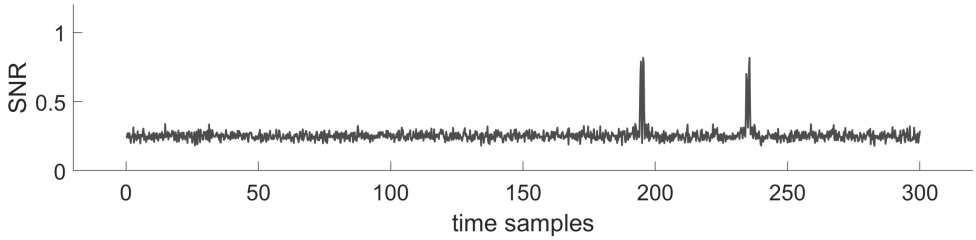
An estimation of this SNR using $q = 1024$ traces, for the same case study as in Chapter 1, is given in Figure 6. It leads to two important observations.

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.

Second, the SNR is different from zero even for non-informative samples, as reflected by the "plateau" value of ≈ 0.25 on the upper and lower plots. This



(a) First S-box.



(b) Second S-box.

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

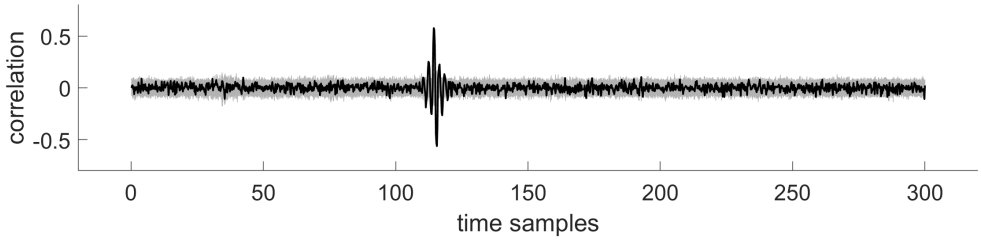
relates to the fact that the SNR is an a priori metric. That is, it defines a priori that any difference in the mean values is signal. When combined with the estimation process of Equation 5, it implies that even estimation errors in the numerator mean values are considered as signal. Such an estimation therefore has a tendency to overestimate the SNR. Quite naturally, "signal" due to estimation errors is not statistically significant and vanishes as q increases, which 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.*)

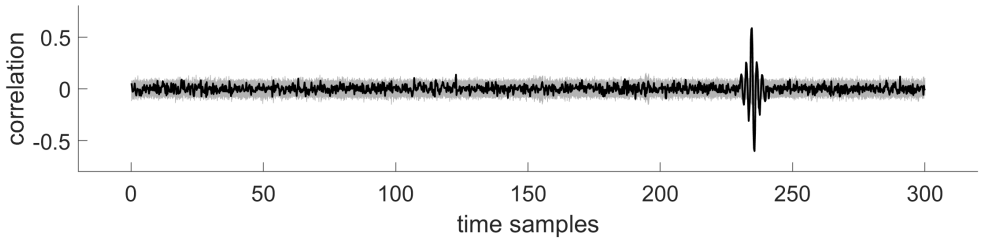
2.2.2 Profiled correlation

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: $m_j^{z_i}(x_i, k_i^*) = \mathsf{HW}\big(\mathsf{S}(x_i \oplus k_i^*)\big)$. 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_{p} profiling traces and q_{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 $\boldsymbol{m}_j^{z_i}(\boldsymbol{x_i}, k_i^*) = \mathsf{HW}\big(\mathsf{S}(\boldsymbol{x_i} \oplus k_i^*)\big)$ 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 detect-

ing 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 ≈ 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_{p} and q_{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 $q_{\mathsf{p}}, q_{\mathsf{a}} \to \infty$ 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]:

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

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^{u} corresponds to a profiled model M_j^{p} with additive model errors M_j^{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 ≈ 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:

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

where $(x_{\mathsf{f}}, k_{\mathsf{f}})$ is a fixed pair of plaintext and key, $\$$ denotes uniformly random values and q_{f} and q_{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.

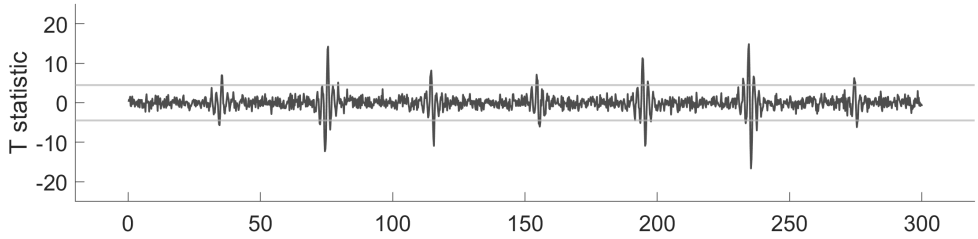


Figure 8: Non-specific T-test for the traces of Figure 1 ($q = 1024$).

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 $(x_\mathsf{f}, k_\mathsf{f})$ 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 $(x_\mathsf{f}, k_\mathsf{f})$ 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 purposes. 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_{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_{a} (fresh) attack traces. For any trace $\boldsymbol{l}^{\iota}$, with $1 \leq \iota \leq q_{\mathsf{a}}$, 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_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 $N_c \leq \tau$ 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:

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

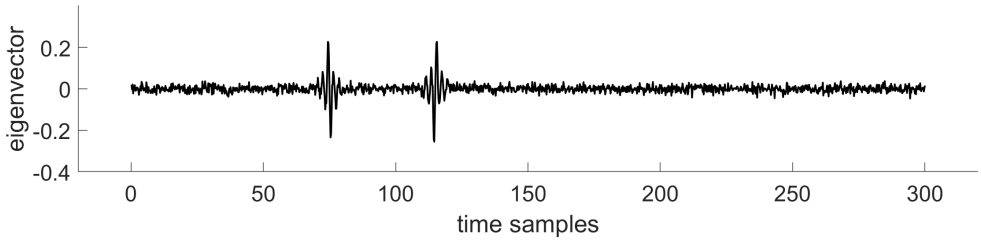
$$\hat{\boldsymbol{\Lambda}}_{z_i} = \hat{\mathsf{E}}_{\frac{q_{\mathsf{P}}}{256}}\Big(\big(\hat{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{L}(z_i) - \hat{\boldsymbol{\nu}}_{z_i}\big) \cdot \big(\hat{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{L}(z_i) - \hat{\boldsymbol{\nu}}_{z_i}\big)^{\mathsf{T}}\Big), \tag{15}$$

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 $\mathsf{f}\big(\boldsymbol{l}^{\iota}|\hat{\boldsymbol{\mu}}_{z_i}, \hat{\boldsymbol{\Sigma}}_{z_i}\big)\big)$ by $\mathsf{f}\big(\hat{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{l}^{\iota}|\hat{\boldsymbol{\nu}}_{z_i}, \hat{\boldsymbol{\Lambda}}_{z_i}\big)\big)$.

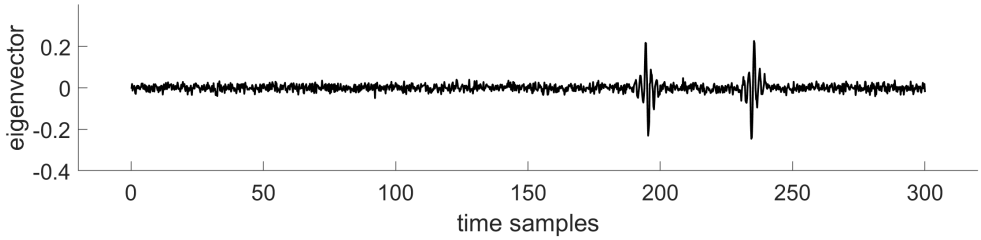
For illustration, the principal direction corresponding to the AES S-boxes' outputs computed with the same simulated traces as previously are given in Figure 9. As expected, the samples selected by PCA match the previously detected ones.

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

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

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

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 $\tilde{\Sigma}_{\mathsf{S}}$ is positive definite and symmetric and since $\tilde{\boldsymbol{w}}$ is scale-invariant, the maximization problem can be replaced by the following eigendecomposition:

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

where $\tilde{\Sigma}_{\mathsf{S}} = \tilde{\boldsymbol{U}}_{\mathsf{S}} \tilde{\boldsymbol{\Delta}}_{\mathsf{S}} \tilde{\boldsymbol{U}}_{\mathsf{S}}^{\mathsf{T}}$ implies $\tilde{\Sigma}_{\mathsf{S}}^{\frac{1}{2}} = \tilde{\boldsymbol{U}}_{\mathsf{S}} \tilde{\boldsymbol{\Delta}}_{\mathsf{S}}^{\frac{1}{2}} \tilde{\boldsymbol{U}}_{\mathsf{S}}^{\mathsf{T}}$ and the projection directions are given by $\tilde{V} = \tilde{\Sigma}_{\mathsf{S}}^{\frac{-1}{2}} \tilde{\boldsymbol{U}}$. 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^{κ}'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 $d \leq 2$, cubic leakage models if $d \leq 3$, … 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 jth time sample l_j^ι 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.

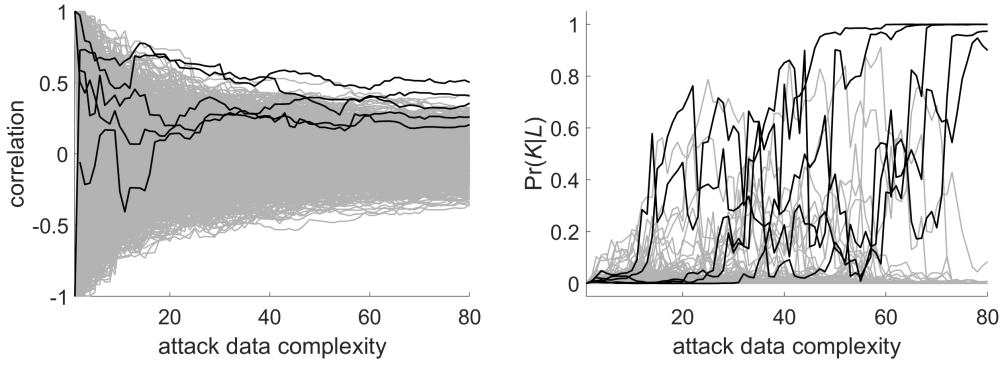
First, and for each information extraction tool, the five attacks do not lead to the same outcome. Such probabilistic results are expected due to the probabilistic nature of side-channel analyses. But it also means that comparing the attack outcomes requires a metric that can cope with random experiments.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

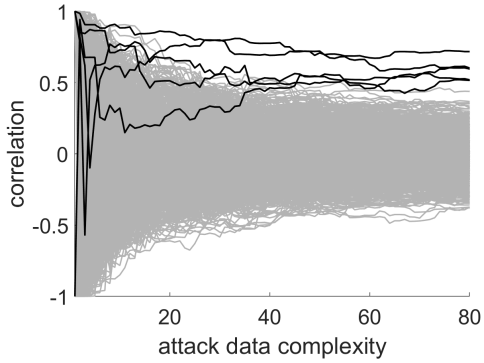
On the other hand, using less leakage assumptions and modeling the leakage of more POIs also has an impact on the profiling complexity of the evaluations. Again, a quantitative comparison will require sound evaluation metrics, of which the definition and discussion is deferred to Chapter 3. Meanwhile, 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.

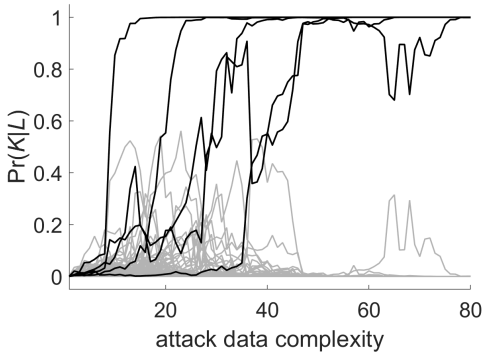


(a) Unprofiled CPA.

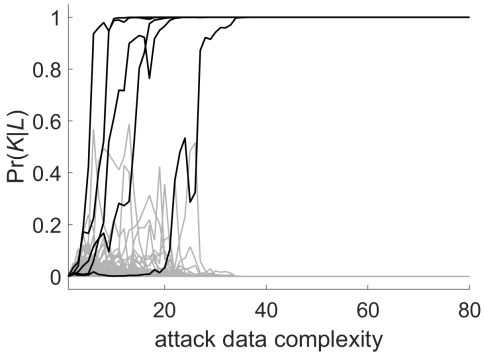
(b) Univatiate LR ($N_b = 9$).



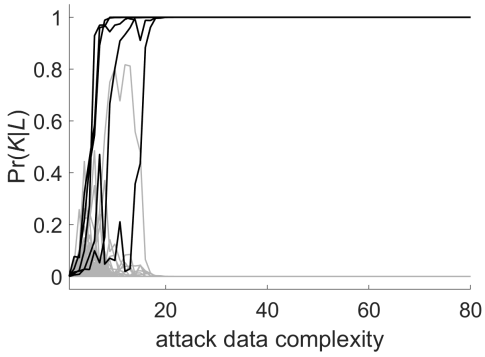
(c) Profiled CPA.



(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).

extraction tool	*assumptions* δ	*noise*	*#POIs exploited*	*profiling data complexity* *# templates*	*size of* Σ	*flavor*
unprofiled CPA	HW	Gaussian + homoscedasticity	one	$\emptyset$	$\emptyset$	metric
profiled CPA	$\emptyset$	Gaussian + homoscedasticity	one	2^η	$\tau \times \tau$	metric
TA	$\emptyset$	Gaussian opt. homoscedasticity	many	2^η	$\tau \times \tau$	probabilistic
PCA-proj. LDA-proj. +TA	$\emptyset$	Gaussian opt. homoscedasticity	many	2^η	$\propto N_c$	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 η-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 256×256 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 16×2^8. If additionally targeting the MixColumn operation, it grows to 4×2^{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.

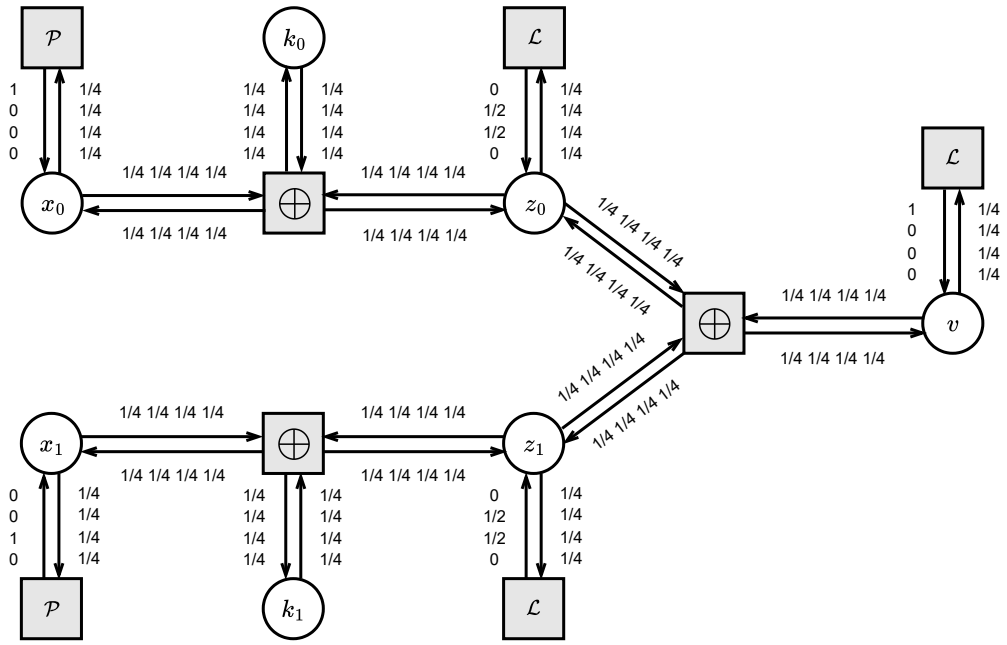


Figure 11: Exemplary factor graph and initialization of a SASCA.

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 α 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.

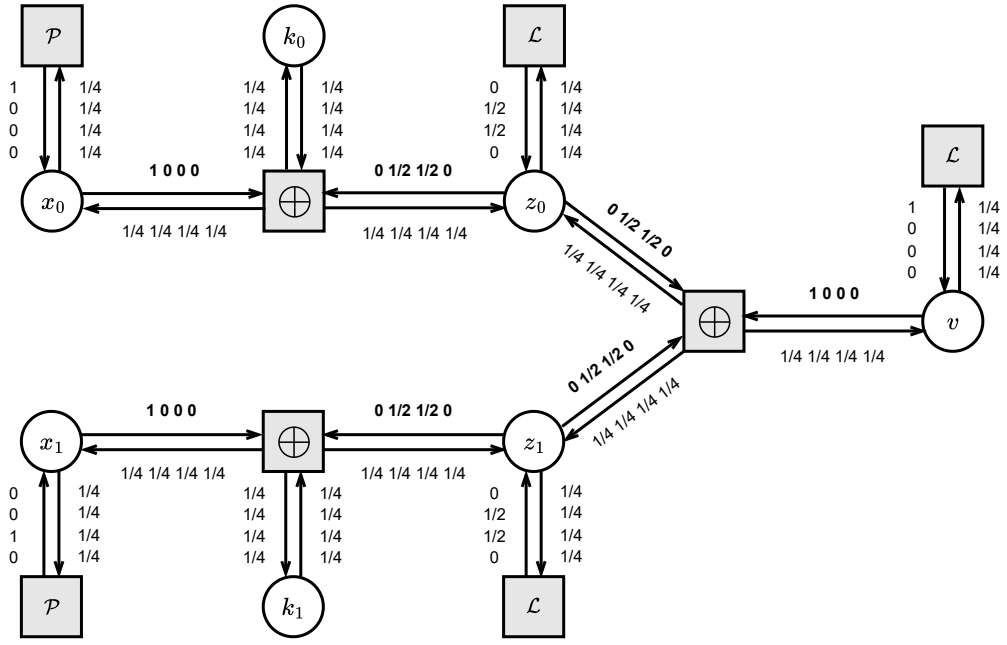


Figure 12: Update of the variable-to-factor messages.

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

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

where $\mathsf{f}(\alpha, \alpha')$ 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 $\mathsf{f}(\alpha, \alpha') = 1$ if $\alpha' = (0, 0)$, which leads to a product of variable-to-factor messages 0 (i.e., $\mathsf{f}(\alpha, \alpha') = 1$ 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 $\alpha = 3$.

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 η-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:

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

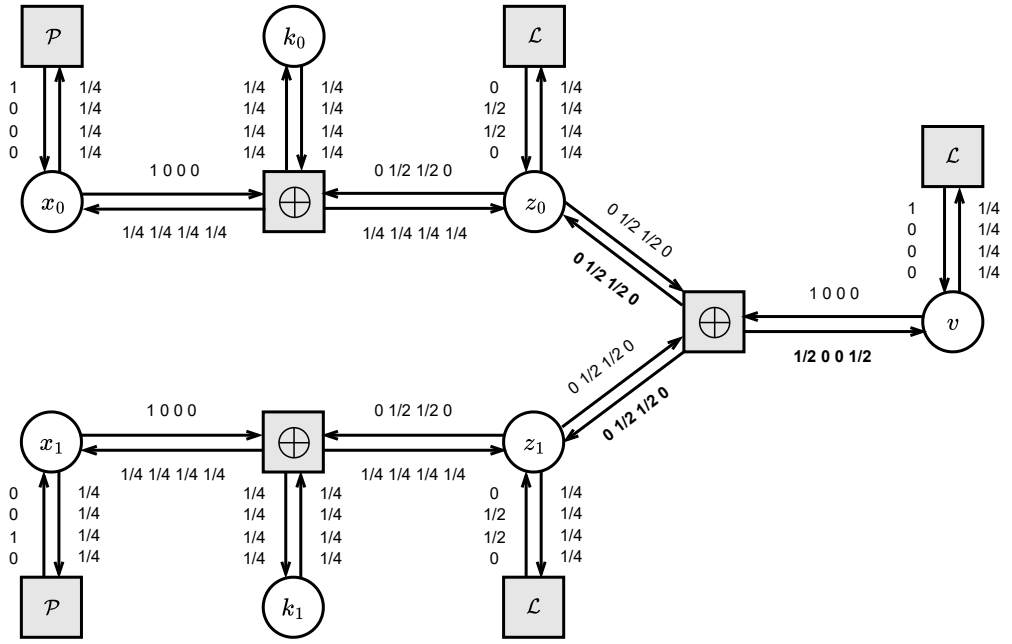
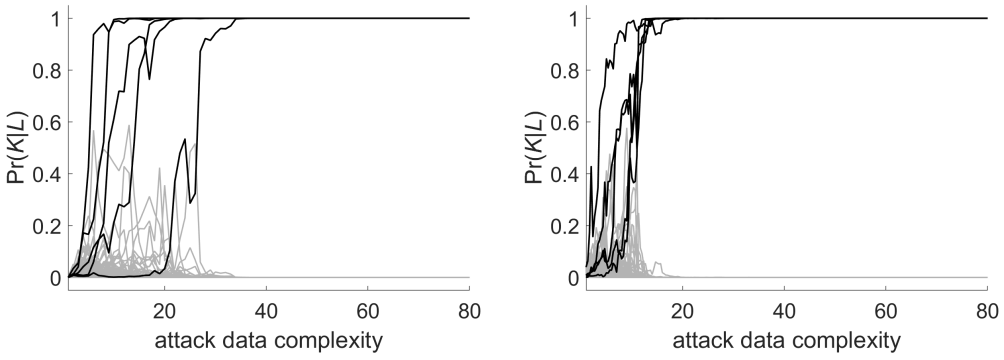


Figure 13: Update of the factor-to-variable messages.

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 $n_c = 4$.

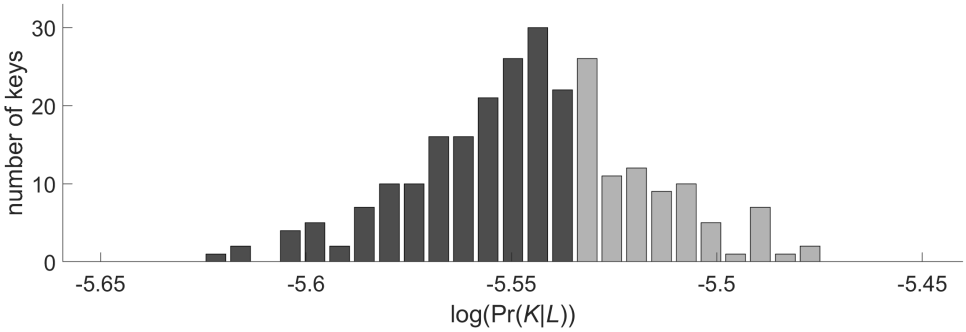
Algorithm 1 only provides an approximation of the key rank, raising the question whether we can bound its errors. For this purpose, we start by explaining the origin of these approximation errors with a 3-bit example. Let us assume that:

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

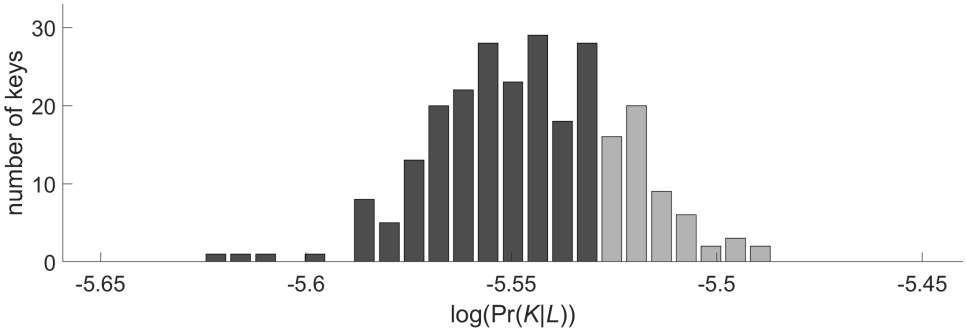
and the correct (concatenated) key is worth $[0, 0]$, so that the first (resp., second) key byte has probability 0.22 (resp., 0.07) after the information extraction and exploitation steps. The corresponding vectors of log probabilities are worth:

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

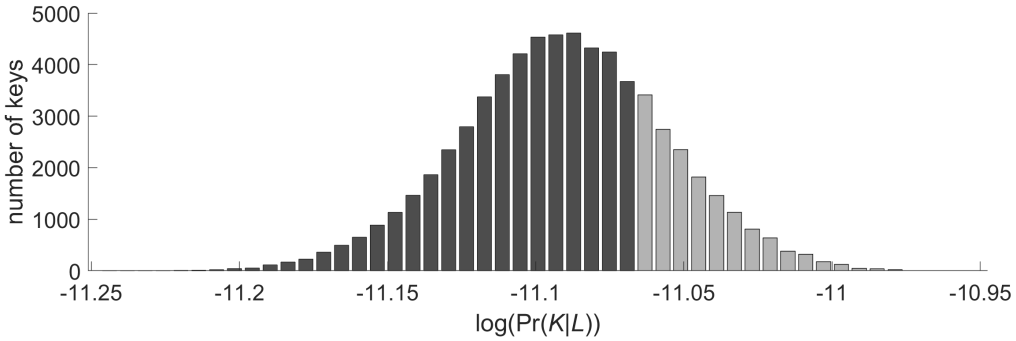
Assuming that $n_b = 3$ bins, it leads to a common bin size of one and histograms with central values $\approx [-4, -3, -2]$ so that we have $\boldsymbol{h}_0 = [2, 1, 5]$ and $\boldsymbol{h}_1 = [0, 4, 4]$. Hence, the convolution $\boldsymbol{h}_0 * \boldsymbol{h}_1$ is a histogram with $N_b = 5$ bins 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:

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

(*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.

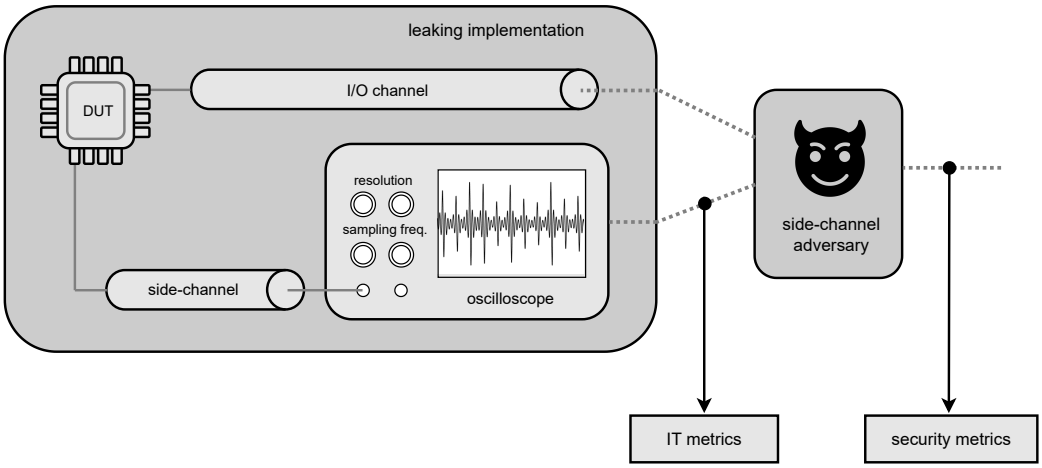


Figure 16: The Eurocrypt 2009 evaluation framework [SMY09].

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 (continous) 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 ℓ 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 range-ℓ 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_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_{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_{a}^{x}, and the number of times the leakage of each plaintext is observed, next denoted as $q_{\mathsf{a}}^{\mathsf{r}}(x)$. 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 can only observe the leakage of a small number of different plaintexts, each of them measured a single time. That is, q_{a}^{x} is bounded and $q_{\mathsf{a}}^{\mathsf{r}} = 1$. A Simple Power Analysis with repetition is a side-channel attack where the adversary can only observe the leakage of a small number of different plaintexts, each of them measured a chosen number of times. That is, q_{a}^{x} is bounded and $q_{\mathsf{a}}^{\mathsf{r}}$ is chosen by the adversary. A Differential Power Analysis is a side-channel attack where the adversary can observe the leakage of a chosen number of different plaintexts. That is, both q_{a}^{x} and $q_{\mathsf{a}}^{\mathsf{r}}$ are chosen by the adversary.*

Figure 17 illustrates the difference between SPA, repSPA and DPA with an example where the leakage caused by the S-box computation of two plaintext bytes x_0^1 and x_0^2 is 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 & DPA).

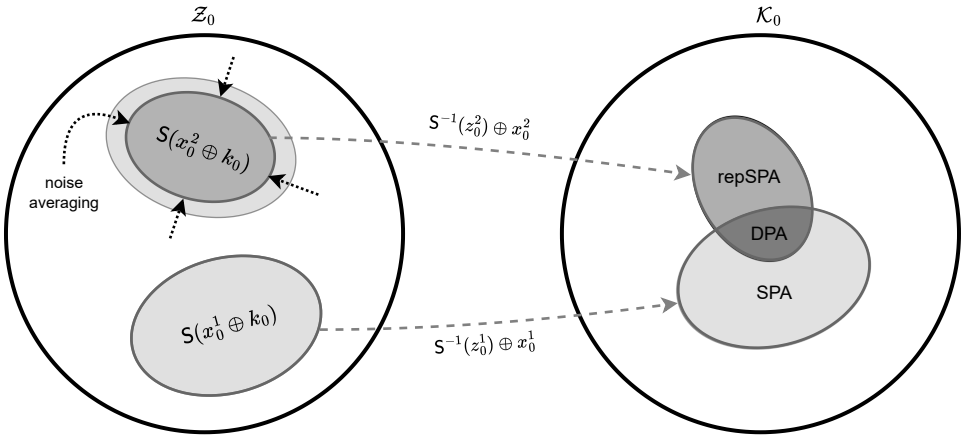


Figure 17: SPA, repSPA and DPA against a single S-box.

We note that this attack taxonomy can be connected with the KIS and KeS variables introduced in Section 2.2.1: KIS variables like the S-box outputs of our example can be targeted by DPA while KeS variables like the ones computed by the AES key scheduling algorithm can only be targeted by repSPA. 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 significantly 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]:

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

The multivariate generalization was finally given by Cherisey et al. [dCGRP19]:

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

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.

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.

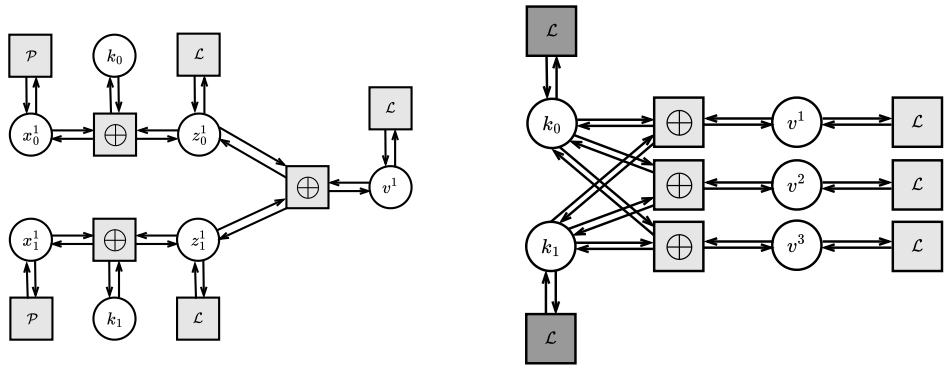
However, it is also important to note that the formulas in this section are all for asymptotic values (i.e., perfect estimations) of the correlation, SNR and MI metrics, motivating our investigation of estimation issues in Section 3.3.

(*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:

$$\underset{v\to f}{\varepsilon} = \sum_{f' \in \text{neighbors(v)} \setminus f} \underset{f'\to v}{\varepsilon}. \tag{35}$$

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:

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

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_{a} plaintexts can be directly set to $q_{\mathsf{a}} \cdot \lambda_Z$ thanks to the (additive) bound of the previous section. Such easy-to-exploit leakages are 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 $\mathcal{L}_\mathsf{p}$:

$$\mathsf{PI}(V;\boldsymbol{L}) = \mathsf{H}(V) + \sum_{v\in\mathcal{V}} \mathsf{Pr}(v) \cdot \int_{\boldsymbol{l}\in\mathcal{L}} \mathsf{f}(\boldsymbol{l}|v) \cdot \log_2 \underbrace{\hat{\mathsf{Pr}}(v|\boldsymbol{l})}_{\text{estimated model}} \; dl, \tag{37}$$

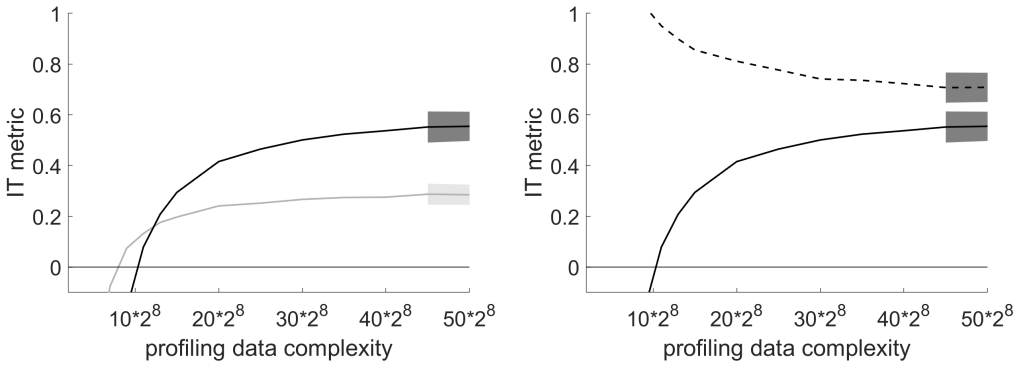
where the hat notation is used to reflect the estimation of the model from which the probabilities are inferred, which may be imperfect (i.e., affected by estimation or assumption errors). This quantity remains difficult to compute since it involves the true leakage distribution $\mathsf{f}(\boldsymbol{l}|v)$ 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:

$$\tilde{\mathsf{PI}}(V;\boldsymbol{L}) = \mathsf{H}(V) + \sum_{v\in\mathcal{V}} \mathsf{Pr}(v) \cdot \underbrace{\sum_{\boldsymbol{l}\in\mathcal{L}_\mathsf{t}(v)} \frac{1}{|\mathcal{L}_\mathsf{t}(v)|}}_{\text{estimated empirical}} \cdot \log_2 \hat{\mathsf{Pr}}(v|\boldsymbol{l}) \; dl, \tag{38}$$

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 (gray) 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 $\mathcal{L}_{\mathsf{p}}^{(\mathrm{j})} = \bigcup_{\mathrm{i}\neq\mathrm{j}} \mathcal{L}_{\mathsf{pt}}^{(\mathrm{i})}$ 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 independent 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 same strategy can be applied to the profiled correlation coefficient of Section 2.2.2.

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:

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

where the overfitted empirical is just the empirical distribution estimated with the profiling set (rather than with the test set as in Equation 38). The TI and the PI of the bivariate TA of our simulated experiments are given in Figure 19b.

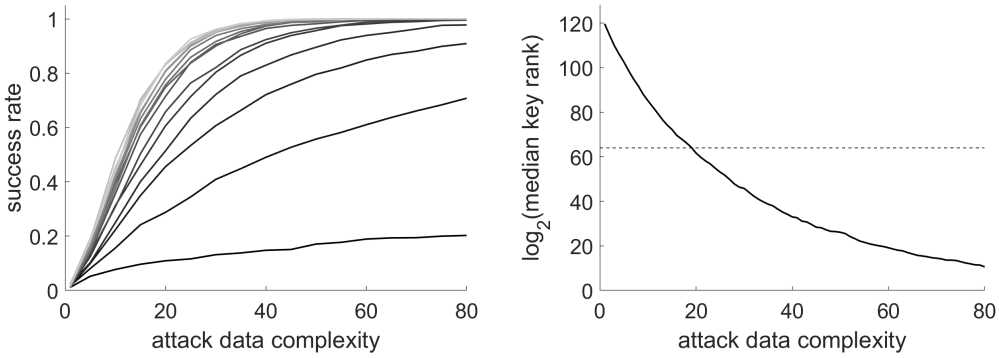
Variability as an additional security margin. We finally note that, in theory, the estimations of this section can be performed with profiling and test sets coming from the same devices or from different devices. As discussed in [BS21], the first option is preferable in a worst-case evaluation setting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

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 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.

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

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



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

Figure 20: Estimation of security metrics.

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 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.

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

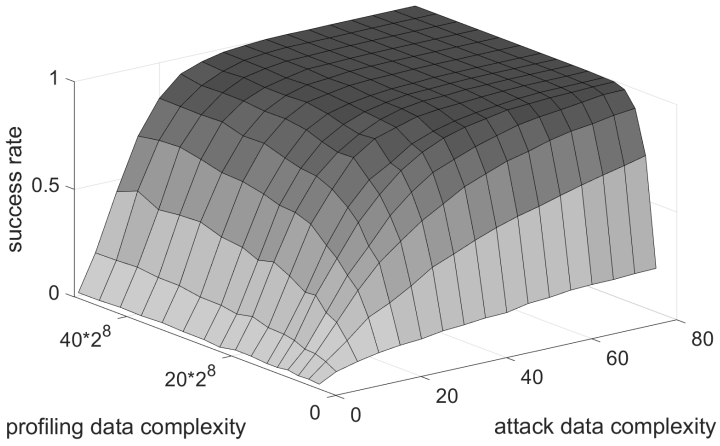


Figure 21: Direct evaluation of the profiling and attack data complexities.

metrics. They should rather be seen as an interesting proxy towards the complete evaluation of an 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 can be more or less optimistic, depending on the target operations and range of success rate targeted. It is for example well-known that for exactly the same leakage model, the range-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 different tools and metrics introduced in this initial 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. In simplified terms, they

observed that generic DPA must leverage an identity leakage model to capture the absence of leakage assumptions and showed the following proposition:

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.*

For our AES case study (where the S-box is bijective), it means that successful side-channel attacks require that the adversary has some intuition about how to make her leakage model surjective. This explains why non-profiled attacks like MIA were shown ineffective with the (generic) identity power model [GBTP08, PR09, VS09, WOM11], or why 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) become 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,

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.

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 22:

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.*

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 countermea-

sures 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.

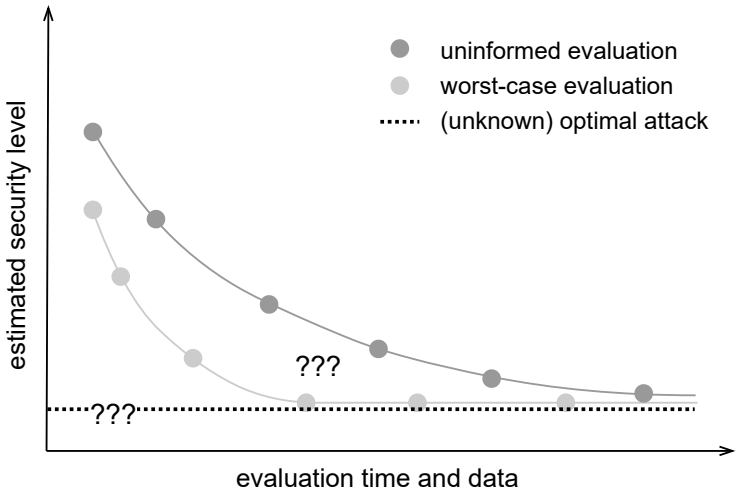


Figure 22: Evaluation conjectures for unprotected implementations.

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

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 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.

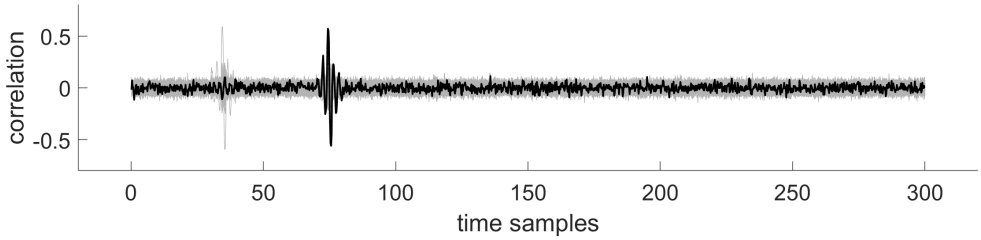
Related works. The separation between information theoretic and security metrics is revisited through the prism of machine learning in [PHJ+19]. The authors make the same

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 (characterizing the S-box) are described in [FLD12]. Rivain analyzes the success rate of the Gaussian side-channel in [Riv08]. Metric-based bounds for the key rank that can be combined with the bounds of Section 3.2 are proposed in [PGS15]. 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]. Finally, 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.

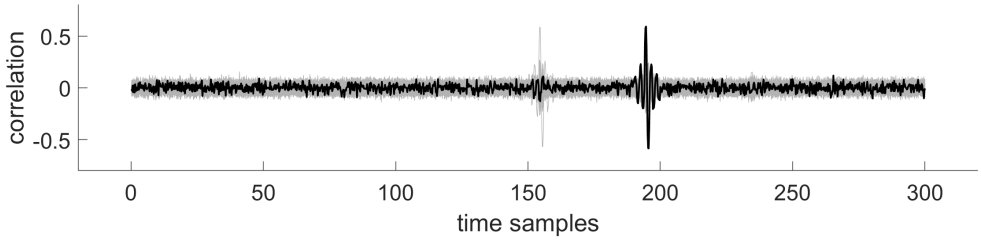
Exercises

Paper-and-pencil

Exercise 1.1 (*Ghost peaks*). Figure 23 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 23: 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 η-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 24. In the 8-bit case (Figure 24a), the adversary receives sixteen leakage samples $l_i^8 = \mathsf{HW}\big(\mathsf{S}(x_i \oplus k_i)\big) + r_i^8$, with $0 \leq i \leq 15$. In the 32-bit case

(Figure 24b), 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 $0 \leq i' \leq 3$. In the 128-bit case (Figure 24c), 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^{s}, and an unpredictable part that corresponds to the other bits manipulated in parallel, which we denote as M_i^{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?

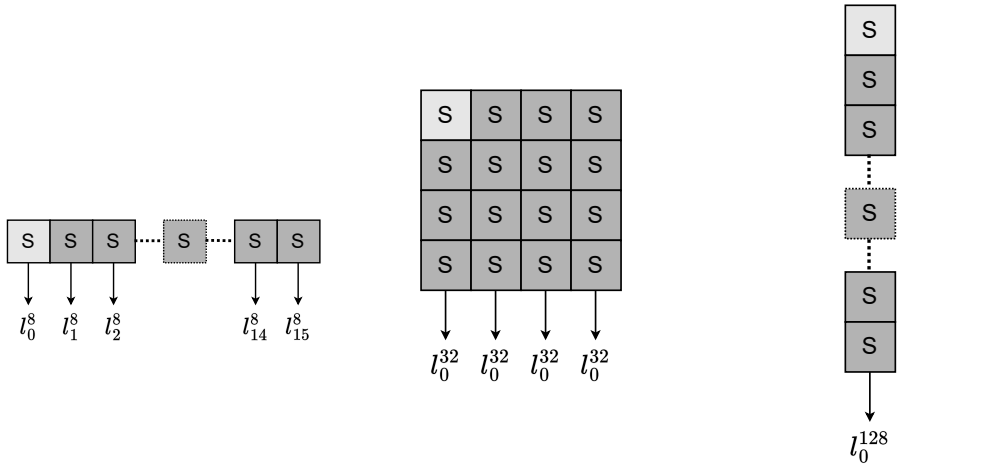
Hint. Use the correlation chain rule of Section 2.2.2 and the following lemma:

Lemma 2 (Maximum correlation for sums of iid variables)**.** *The correlation coefficient between the sum of η 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 η-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 24c 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 24: 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 24b 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 λ_Z and λ_V are identical.

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 (`https://www.simple-crypto.org/activities/scale/`).

Part II

Leakage-Resistance

Introduction

Chapter 4

Security definitions

4.1 Completing the attacks' taxonomy

4.2 Integrity with leakage

4.3 Confidentiality with leakage

4.4 Authenticated encryption

Chapter 5

Mode-level and primitive-level countermeasures

5.1 Leakage-resistant MACs

5.2 Leakage-resistant encryption

5.3 Leakage-resistant AE

5.4 Primitive-level optimizations

Chapter 6

Implementation-level countermeasures

6.1 Masking basics

6.2 Shuffling

6.3 Combining countermeasures

6.4 Masking theory

Exercises

Part III

Secure Implementations

Introduction

Chapter 7

Hardware designs

7.1 Ensuring DPA security

7.2 Ensuring SPA security

Chapter 8

Software designs

8.1 Ensuring DPA security

8.2 Ensuring SPA security

Chapter 9

Backwards security evaluations

9.1 Testing DPA security

9.2 Testing SPA security

9.3 Discussion and related works

Exercises

Concluding remarks

Open problems and related works

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. As for the post-processing step, it is mostly the efficient enumeration of dependent secrets that would benefit from improvements. These last difficul-

ties are in turn reflected in the theoretical background of Chapter 3, where the bounds and shortcuts for adaptive or analytical attacks and the integration of the adversary's computational (enumeration) power remain missing or heuristic.

The link to hardware countermeasures

Bibliography

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

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

[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.

[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.

[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.

[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.

[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.

[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.

[DDF19] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. *J. Cryptol.*, 32(1):151–177, 2019.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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 pipelining 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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[YEM14] Xin Ye, Thomas Eisenbarth, and William Martin. Bounded, yet Sufficient? How to Determine Whether Limited Side Channel Infor-

mation 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.

[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 25 where the x_i's denote the sixteen plaintext bytes and k_i^ϱ denotes the byte i of the expanded key at round ϱ. 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 4, 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 26:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \text{'02'} & \text{'03'} & \text{'01'} & \text{'01'} \\ \text{'01'} & \text{'02'} & \text{'03'} & \text{'01'} \\ \text{'01'} & \text{'01'} & \text{'02'} & \text{'03'} \\ \text{'03'} & \text{'01'} & \text{'01'} & \text{'02'} \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}, \tag{40}$$

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:

$$\big(b(6), b(5), b(4), b(3) \oplus b(7), b(2) \oplus b(7), b(1), b(0) \oplus b(7), b(7)\big). \tag{41}$$

AddRoundKey is a 128-bit bitwise XOR between the state and the round key.

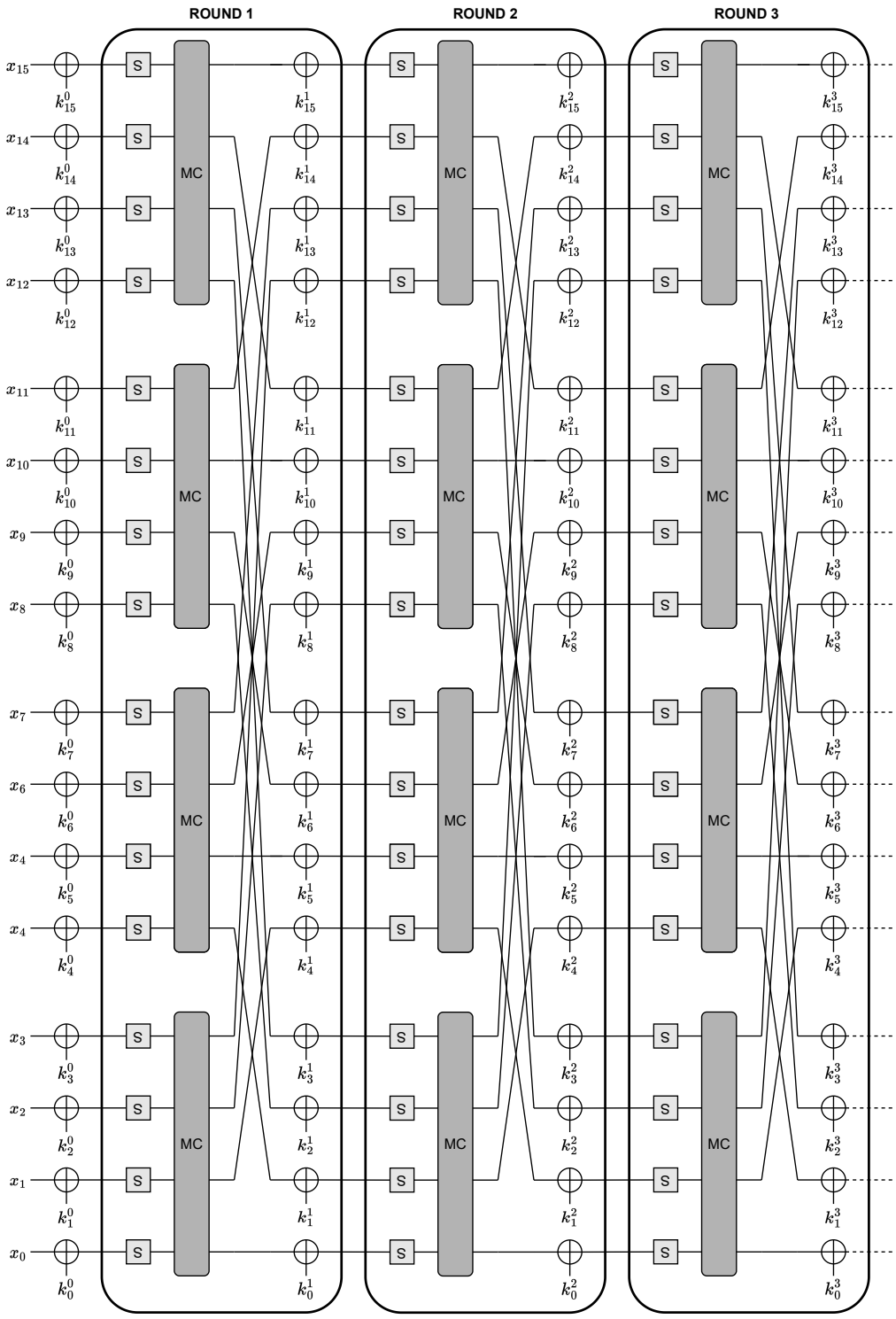


Figure 25: Three first rounds of the AES Rijndael (wires are bytes).

$b14 = a3 \oplus a5$	$b1 = c0 \oplus a7$	$b15 = c1 \oplus a5$	$b17 = b10 \oplus b11$
$b13 = a0 \oplus a6$	$b4 = b1 \oplus a3$	$b20 = c1 \oplus a1$	$b19 = b10 \oplus b8$
$b12 = b13 \oplus b14$	$b2 = b1 \oplus a0$	$b6 = b15 \oplus a7$	$b16 = c0 \oplus b11$
$b9 = a0 \oplus a3$	$b5 = b1 \oplus a6$	$b10 = b15 \oplus c0$	$b21 = b13 \oplus b16$
$b8 = a0 \oplus a5$	$c1 = a4 \oplus b12$	$b11 = b20 \oplus b9$	$b18 = a0 \oplus b16$
$c0 = a1 \oplus a2$	$b3 = b5 \oplus b8$	$b7 = a7 \oplus b11$	

(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 4: Bitslice AES S-box from [BMP13].

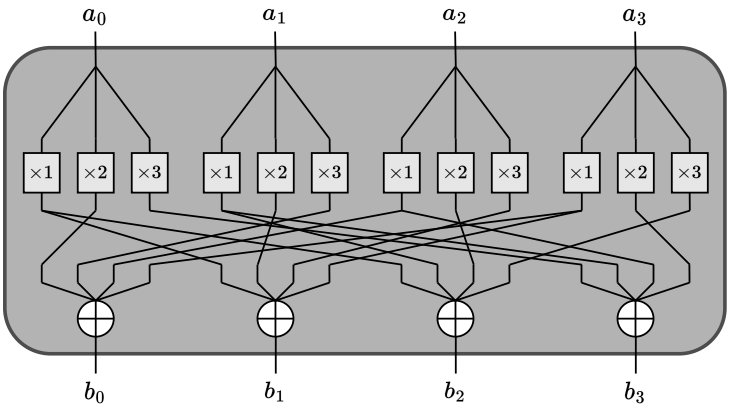


Figure 26: The AES MixColumns operation (wires are bytes).

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.