

Towards Fair Side-Channel Security Evaluations

François DURVAUX

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

Composition du jury :
Pr. David Bol (UCL) – Président
Pr. Jean-Didier Legat (UCL)
Pr. Olivier Pereira (UCL) – Secrétaire
Dr. Axel Poschmann (NXP)
Pr. François-Xavier Standaert (UCL) – Promoteur
Dr. Nicolas Veyrat-Charvillon (CAIRN-IRISA)

Louvain-la-Neuve, Belgique – 2015

Abstract

Side-channel attacks appeared for the first time in the late 90's. They rely on the observation that the physical features of a cryptographic device may reflect its internal activity which may reveal sensitive information such as encryption keys. This unintended leakage is hardly controlled, and in general cannot be totally prevented. Therefore, determining the true security level given these leakages is an important open problem in modern cryptography. In order to provide worst-case security guarantees, the evaluator needs to accurately model the leakages. Yet, in practice, various issues may be encountered and make this task challenging. For example: (i) the sensitive data is generally processed at different times, hence the leaked information is spread in the measurements, (ii) the leakage model may be biased, in which case a part of the leaked information is missed. In this thesis, we aim to contribute to the fair evaluation of cryptographic devices in three directions: (1) the leakage detection, (2) the detection of Points-Of-Interest (POIs), and (3) the leakage certification. The leakage detection determines if data-dependent leakages are present in the measurements, independent of whether they can be exploited. By contrast, the POI detection identifies the samples that can be used to recover the secret key. In the first part of the thesis, we investigate these two tasks and put forward that while having different purposes, they are also connected to a significant extent. We also propose concrete improvements for both, and show how to exploit heuristic optimization algorithms to improve the POI detection for implementations protected by side-channel attack countermeasures. In the second part of the thesis, we introduce leakage certification methods in order to test the quality of the evaluator's model. We show how the sources of error can be separately identified and quantified. Moreover, we show that the underlying information loss can be bounded.

Acknowledgements

The first person I would like to warmly thank is my supervisor, François-Xavier Standaert. Without him, nothing in this thesis would have been possible. Thanks for giving me this opportunity, and for being so interested, involved, and supportive (and for all the “kicks in the arse” when I deserved them). One could not hope a better environment for completing such a long (and sometimes painful – I don’t particularly enjoy writing) personal achievement that is the PhD thesis. I would also like to thank all the members of my jury for their advice and their support: Nicolas Veyrat-Charvillon, Olivier Pereira, Jean-Didier Legat, David Bol, and Axel Poschmann.

If it was not for the passion for the scientific research, I went to work everyday with enthusiasm, and I owe this to the people of the Crypto Group. I thank them all for their friendship and the moments we shared. I would like to specially mention my French mates, Vincent and Romain, for all the bars we roamed together at international conferences; Sylvie for her constant smile, her support, and for having the patience to read me; Loïc for all the deep philosophical conversations; and of course my former colleague and friend Stéphanie who co-authored many of my research papers.

Next, I must thank the “Team”. They are all these lads with whom I had some good years in Louvain-la-Neuve: JB, Ka, Jey, Xa, Sam, Nico, Romain, GBB, JC, and Dani. Having this experience together was awesome. I want to thank the “MLT Crew”: the guys who have been present since I was a kid, and have always supported me. They are Jean, Xa, Tony, Ben D, and Ben C. I am grateful to my parents, Anne and Marc, my brothers, David and Xavier, and the rest of my family (with a special mention to Dani, Pypy and Oli) for being so interested and supportive. Finally, the last but definitely not the least, I deeply thank Nathalie, my life’s companion. She has always been there to support me, help me, and cheer me up, no matter what.

I am grateful to the Walloon Region (with the MIPSs project) and the European Commission (with the ERC CRASH project) for funding my research during these four years and a half.

Contents

1	Introduction	1
2	Background	7
2.1	AES on microcontrollers	7
2.1.1	Iterated block ciphers	8
2.1.2	Advanced Encryption Standard	10
2.1.3	Software implementations	11
2.1.4	Case-study: AES Furious	15
2.2	Side-channel analysis	16
2.2.1	Power traces	17
2.2.2	Standard DPA	19
2.2.3	Countermeasures	22
2.2.4	Eurocrypt 2009 framework	26
2.3	Instantiation	27
2.3.1	Measurement setup	27
2.3.2	PDF estimation methods	28
2.3.3	Metrics	30
2.3.4	Estimating a metric with cross-validation	32
2.4	SCA workflow	33
2.4.1	Fixed vs. random leakage detection test	35
2.5	Conclusion	36
3	Leakage and points-of-interest detection	37
3.1	Understanding the leakage detection	39
3.1.1	A correlation-based leakage detection test	41
3.1.2	Comparison with the <i>fixed vs. random</i> test	42
3.1.3	Improved leakage detection test	48
3.1.4	Summary & open problems	51
3.2	Selecting time samples with projection pursuits	53
3.3	PPs against unprotected devices	55
3.3.1	Projection pursuit algorithm	56

3.3.2	Experimental results	57
3.3.3	Discussion & comparison with existing works . . .	59
3.4	PPs against masked implementations	59
3.4.1	Specialized projection pursuit algorithm	60
3.4.2	Simulated experiments	64
3.4.3	Measured experiments	66
3.4.4	Discussion & comparison with existing works . . .	68
3.4.5	Conclusion	71
3.5	From leakage detection to POI detection	72
3.5.1	The ρ -test as a POI detection tool	73
3.5.2	Experimental validation	74
3.5.3	Conclusion	75
4	Leakage certification	79
4.1	How to certify the leakage of a chip?	81
4.1.1	Estimation errors and cross-validation	82
4.1.2	Assumption errors and distance sampling	84
4.1.3	Experimental results	87
4.1.4	Estimation vs. assumption errors	88
4.1.5	Can we bound the information loss?	91
4.1.6	Pragmatic evaluation guidelines & conclusions . . .	95
4.2	Towards easy leakage certification	97
4.2.1	A motivating negative result	98
4.2.2	A new method to detect assumption errors	99
4.2.3	Simulated experiments	102
4.2.4	Measured experiments	103
4.2.5	Quantifying the information loss	105
5	Other work	111
5.1	Hardware implementations	111
5.2	Intellectual property protection	114
5.3	Side-channel attacks	116
5.4	Internship at Cryptography Research Inc.	117
6	Conclusion and perspectives	119
	References	125

List of notations

General notations

X	a random variable
x	a realization of the random variable X
$\mathcal{X}$	the set of all possible values of X
$f(X), M(X)$	functions of the random variable X
$\hat{x}, \hat{f}$	an estimated variable or function
$\mathbf{X}, \mathbf{x}$	a vector of random variables or realizations
$\Pr[X = x]$	probability of X taking the value of x
$\Pr[X = x Y = y]$	conditional probability of $X = x$ if $Y = y$

Cryptographic notations

p, c, κ	the plaintext, ciphertext, and secret key
x, s	the target plaintext and subkey bytes
X, S	random variables corresponding to the target bytes
S	a S-box function (substitution box)
y, z	AddRoundKey and SubBytes outputs, given x and s
N_s	number of time samples in a leakage trace
τ	time sample index, $\tau \in [0; N_s - 1]$
$l_y(\tau)$	a sample of a leakage trace, at time τ and given y
$\mathcal{L}$	a set of leakage traces

Abbreviations

AES	Advanced Encryption Standard
CDF	Cumulative Density Function
CPA	Correlation Power Analysis
DPA	Differential Power Analysis
GT	Gaussian Template
HD	Hamming Distance
HW	Hamming Weight
LDA	Linear Discriminant Analysis
LR	Linear Regression
MCP-DPA	Moments-Correlating Profiled DPA
MI	Mutual Information
MMPC	Moments against Moments Profiled Correlation
PCA	Principal Component Analysis
PDF	Probability Density Function
PI	Perceived Information
POI	Point-Of-Interest
PP	Projection Pursuit
SCA	Side-Channel Analysis
SNR	Signal-to-Noise Ratio
SPA	Simple Power Analysis
TA	Template Attack

List of publications

- St  phanie Kerckhof, Fran  ois Durvaux, Nicolas Veyrat-Charvillon, Francesco Regazzoni, Gu  rric Meurice de Dormale, and Fran  ois-Xavier Standaert. Compact FPGA implementations of the five SHA-3 finalists. In Emmanuel Prouff, editor. *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, volume 7079 of *Lecture Notes in Computer Science*. Springer, 2011., pages 217–233.
- Fran  ois Durvaux, Beno  t G  rard, St  phanie Kerckhof, Fran  ois Koeune, and Fran  ois-Xavier Standaert. Intellectual property protection for integrated systems using soft physical hash functions. In Dong Hoon Lee and Moti Yung, editors, *Information Security Applications - 13th International Workshop, WISA 2012, Jeju Island, Korea, August 16-18, 2012, Revised Selected Papers*, volume 7690 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.
- David Bol, Julien De Vos, C  dric Hocquet, Fran  ois Botman, Fran  ois Durvaux, Sarah Boyd, Denis Flandre, and Jean-Didier Legat. A 25mhz 7  w/mhz ultra-low-voltage microcontroller soc in 65nm LP/GP CMOS for low-carbon wireless sensor nodes. In *2012 IEEE International Solid-State Circuits Conference, ISSCC 2012, San Francisco, CA, USA, February 19-23, 2012*, pages 490–492. IEEE, 2012.
- St  phanie Kerckhof, Fran  ois Durvaux, C  dric Hocquet, David Bol, and Fran  ois-Xavier Standaert. Towards green cryptography: A comparison of lightweight ciphers from the energy viewpoint. In Emmanuel Prouff and Patrick Schaumont, editors. *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012*.

Proceedings, volume 7428 of *Lecture Notes in Computer Science*. Springer, 2012. , pages 390–407.

- François Durvaux, Mathieu Renaud, François-Xavier Standaert, Loïc van Oldeneel tot Oldenzeel, and Nicolas Veyrat-Charvillon. Efficient removal of random delays from embedded software implementations using hidden markov models. In Stefan Mangard, editor. *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers*, volume 7771 of *Lecture Notes in Computer Science*. Springer, 2013. , pages 123–140.
- Stéphanie Kerckhof, François Durvaux, François-Xavier Standaert, and Benoît Gérard. Intellectual property protection for FPGA designs with soft physical hash functions: First experimental results. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013, Austin, TX, USA, June 2-3, 2013*, pages 7–12. IEEE Computer Society, 2013.
- Ludovic-Henri Gustin, François Durvaux, Stéphanie Kerckhof, François-Xavier Standaert, and Michel Verleysen. Support vector machines for improved IP detection with soft physical hash functions. In Emmanuel Prouff, editor. *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*. Springer, 2014. , pages 112–128.
- François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. How to certify the leakage of a chip? In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 459–476. Springer, 2014.
- 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. To appear in the proceedings of *COSADE 2015*, 2015.
- François Durvaux and François-Xavier Standaert. Towards easy leakage certification. Currently submitted, 2015.

- François Durvaux and François-Xavier Standaert. From leakage detection to the detection of points of interests in leakage traces. Currently submitted, 2015.

Chapter 1

Introduction

Cryptography is the science of communicating in the presence of an adversary. This engineering field was originally dedicated to a military use. The history keeps tracks of famous devices such as the Caesar's cipher or the Enigma machine used by German forces during the Second World War. Their purpose was to exchange strategic plans that would remain secret, even though the message was intercepted by a foe. More recently, it has gained a certain interest for civilian applications with the expansion of communication networks such as the Internet. Tremendous amounts of our personal data are manipulated by these means and this is becoming even more important with the ongoing progress of the Internet-of-Things (IoT): every small object of our everyday life will be connected. Sensitive information about our lives are collected and communicated through these networks: bank account details, personal and business relationships, hobbies, family pictures, health status, and a lot more. It usually makes our lifestyle more convenient but it is at the risk of losing control on how and where this information is propagated, and for what purpose. Therefore, communication devices generally make use of cryptographic primitives (e.g. for encryption, authentication, and signature) in order to protect our personal data against outside threat. However, even though these primitives are proved to be secure, newspapers frequently relate stories of huge security flaws that are revealed, often to the disadvantage of the users. In fact, the overall security of the device may suffer from unintended implementation side-effects. In this context, providing independent and fair security evaluations is an urgent matter. It is one of the biggest challenges the research community in cryptography is currently facing. Evaluating the side-channel security of cryptographic primitives is a small step in this direction.

In general, while cryptographic algorithms are designed to be theoretically secure against hypothetical adversaries with high computational power, their physical realization – their implementation – may contain major security flaws that are independent of their mathematical properties. In this thesis, we focus on Side-Channel Analysis (SCA), introduced in the late 90’s by Paul Kocher. Under the assumption that the adversary has a physical access to the device, it is possible for him to monitor (and possibly tamper with) its physical features that depend on the algorithm execution (operation and data). With the appropriate statistical tools, he can thereby extract information about the sensitive values, including the secret key. Since SCAs were first described, they have proved to be a real threat to embedded devices. An evaluator must consider this type of attacks in order to provide a realistic security evaluation of the systems.

In this context, the security level directly depends on the amount of information contained in the measurements taken by the adversary, and his ability to exploit it. The measurements can be seen as a series of noisy time samples, each corresponding to a given instant of the algorithm execution. Some contain useful information for attacking (denoted as signal), others do not. Protections against side-channel attacks do exist. Among the wide variety of countermeasures that are proposed in the literature, masking is one of the most challenging to deal with for an attacker. It consists in mixing intermediate values with random values, called masks. Concretely, it has the effect of splitting the sensitive information into *shares* that are spread on different time samples in the measurements. This technique increases the data complexity of a potential attack: more traces (i.e. measurements) are needed to retrieve the sensitive information hidden in higher statistical moments. It also generally increases the complexity of the preprocessing the adversary needs to perform prior to an attack, as he must find the useful points first. Indeed, they must be treated simultaneously, otherwise no information is visible.

Nowadays, security evaluations are performed by certification labs. The question they aim to answer is twofold: (i) “Is there any leaking information?” and (ii) “How many traces are needed to recover the key?”. The first question relates to the leakage detection and Points-Of-Interest (POI) detection problems, which are done as preliminary steps before attacking. Leakage detection usually refers to the task of identifying data-dependent information in side-channel measurements, independently of whether this information can be exploited. Detecting POIs in leakage traces is a complementary task that is a necessary first

step in most side-channel attacks where the adversary wants to turn this information into (e.g.) a key recovery. Yet, it is a frequently neglected problem in the side-channel literature, even though it can become the bottleneck of practical adversaries/evaluators as the size of the measurement traces increases. This is particularly true in the challenging context of masked implementations where only a combination of the informative samples reveals exploitable information. The second question is a leakage certification problem. Evaluating side-channel attacks and countermeasures requires determining the amount of information leaked by a target device. For this purpose, information extraction procedures published so far essentially combine a “*leakage model*” with a “*distinguisher*”. A leakage model is an approximation of how the sensitive data is reflected in the physical feature of the device. The distinguisher is generally a statistic that identifies the best key candidates, i.e. for which the leakage model best fits the observations. Fair evaluations ideally require exploiting a perfect leakage model (i.e. exactly corresponding to the true leakage distribution) with a Bayesian distinguisher. But since such perfect models are generally unknown, density estimation techniques have to be used to approximate the leakage distribution. This raises the fundamental problem that all security evaluations are potentially biased by both estimation and assumption errors. Estimation errors appear if not enough samples are used to estimate the considered model, while the assumption errors appear if the model does not correspond to the actual observations. Hence, the best that we can hope is to be aware of these errors.

In this thesis we consider these aspects and aim to provide a fair side-channel security evaluation for embedded devices. For this purpose, we tackle the two aforementioned questions in different parts of the reported work:

(1) We discuss the connections between the leakage detection and POIs detection tasks, by investigating the differences between a popular solution to leakage detection based on a t-test (currently widely used by certification labs), and an alternative method exploiting Pearson’s correlation coefficient. We first show that the simpler t-test can have better sampling complexity, and that its gain over the correlation-based test can be predicted by looking at the signal-to-noise ratio of the leakage partitions used in these tests. This implies that the sampling complexity of both tests relates to their implicit leakage assumptions, rather than the actual statistics exploited. We then show that this gain comes at the cost of some intuition loss regarding the localization of the exploitable leakage samples in the traces, and their informativeness. Therefore,

whereas t-tests are the method of choice for leakage detection only, correlation-based tests exploiting larger partitions are preferable for detecting POIs. In parallel, we describe new (black box) tools for efficiently selecting the time samples of masked implementations. The proposed techniques exploit projection pursuits and specialized local search algorithms, work with minimum memory requirements and practical time complexity. We validate them with two case-studies of unprotected and first-order masked implementations in an 8-bit device, the latter one being hard to analyze with previously known methods. Next, we confirm the intuition drawn from our leakage detection investigation by integrating a correlation-based leakage detection test in the automated tool for the detection of POIs in masked implementation traces.

(2) Considering the estimation and assumption errors that may appear in the certification process, we provide and implement methodological tools to identify and quantify them. First, we show how sound statistical techniques allow both quantifying the leakage of a chip, and certifying that the amount of information extracted is close to the maximum value that would be obtained with a perfect model. Side-channel attacks generally rely on the availability of good leakage models to extract sensitive information from cryptographic implementations. The proposed tool aims to guarantee that this condition is fulfilled based on sound statistical arguments. They are important ingredients in the evaluation of leaking devices since they allow a good separation between engineering challenges (how to produce clean measurements) and cryptographic ones (how to exploit these measurements). Next, we propose an alternative leakage certification test that is significantly simpler to implement. This gain admittedly comes at the cost of a couple of heuristic (yet reasonable) assumptions on the leakage distribution. To confirm its relevance, we first show that it allows confirming our first (more formal) results of leakage certification. We then put forward that it leads to additional and useful intuitions regarding the information losses caused by incorrect assumptions in leakage modelling.

Main contributions

During our investigations, many efforts were made in order to provide a better understanding of the current security evaluation process, and to enhance the tools provided in the literature. This led to the following contributions:

- **Practical insight.** In the research, we investigate the practical challenges that are encountered by an evaluator. We also analyze

the tools that are generally used by most certification labs. Concretely, we study the connection between the leakage detection and POI detection problems, which is not really discussed in the literature. For that purpose, we also analyse the functioning of a widely used tool based on a t-test and underline its pros and cons. This analysis also allows us to propose a tweaked version of the test that provides better detection results. Regarding the certification problem, we show that the different sources of errors can be identified, separated and treated independently. We also show how the results provided by these techniques are linked to the information theoretic and security metrics provided in the literature.

- **New tools.** In the POI detection part of this work we provide a new method and algorithm in order to find the informative points in masked implementation traces. This method allows us to gain a constant – but practically significant – factor in the search speed, in comparison to all the methods that were previously proposed. Regarding the certification process, we propose new methods that allow the evaluator to estimate the quality of his leakage model, which is a premiere in the context of side-channels.
- **New application of existing tools.** In this thesis we also extend the use of some already existing tools in order to complete our investigations. For instance, we show how the well-known Pearson’s correlation can be used in order to answer various questions such as leakage detection, high-order POI detection, and quantifying the information loss (if any) during a certification process. We also show how the t-test, that is originally used for leakage detection, can be extended to assumption error detection. Finally, we introduce the use of methods such as the cross-validation (widely used in machine learning) or the local search algorithms family in the context of side-channel, in order to provide new solutions and best existing results.
- **Investigations in practical scenarios.** Applying signal processing methods and new attacks on actual measurements is not always an easy thing to do. In this thesis, we always confirm our claims and our proposed approaches on actual leakage measurements. In other words, we show how our results can effectively be reproduced in real world evaluations with reasonable and practical assumptions.

To sum up, in this thesis we provide a complete set of tools that simplify side-channel security evaluations of embedded devices. The

proposed methods are designed such that they are as independent as possible from any evaluator's assumption. They are a first step towards fair side-channel security evaluations which, we believe, will open doors to future research in that direction.

Chapter 2

Background

The work reported in this thesis focuses on providing fair side-channel security evaluations. We first investigate statistical tools and heuristics for a better understanding of side-channel leakages. We then provide methods and statistics for (i) enhancing the leakage and points-of-interest detections and (ii) evaluating the soundness of statistical models. All our experiments are performed on an 8-bit implementation of the Advanced Encryption Standard (AES). This background section places the context of this work and introduces the tools and concepts it is based on. A first overview of the structure of iterated block ciphers is given. AES and its software implementation for embedded applications is then described. Next, the concept of Side-Channel Analysis (SCA) and standard DPA are introduced, and an overview of existing side-channel countermeasures is given. Details are provided for a pre-computed masking scheme whose implementation is used to validate our work on points-of-interest detection. We refer to the Eurocrypt 2009 framework in order to define the metric we use to interpret our results on a fair basis. Next, the instantiations of our setup and the metrics we use are described. The literature provides all kind of statistics for evaluating the side-channel security of cryptographic implementations. We list the ones that best apply to our context and that we further investigate in this work. Finally, an informal description of a practical side-channel analysis workflow is given and the non-specific leakage detection test is introduced.

2.1 AES on microcontrollers

The reasoning and the experiments in the next chapters are based on the AES in the context of embedded devices, microcontrollers in particular. For this purpose, this first section introduces the cryptographic

algorithm family that is the iterated block ciphers. Then, the AES instantiation is presented. We next present the microcontroller architectures and their programming specificities. The exact implementation used for all the experimental validations made along this work is eventually described.

2.1.1 Iterated block ciphers

Cryptography is a very wide topic that involves lots of different engineering skills. It is intended for applications that require some of the following properties: confidentiality (hiding sensitive information), data integrity (preventing data from being altered by an unauthorized party), and authentication (verifying user's identity, which generally implies the integrity) [52]. In this thesis, we illustrate our claims with the confidentiality property. For this purpose, cryptographic algorithms are divided into two main families: *symmetric* and *asymmetric* algorithms. The former refers to algorithms that use the same *key* for encryption and decryption (respectively hiding and recovering the sensitive data). The latter refers to algorithms that respectively use a *public* key (known by everyone) and a *private* key (only known by the recipient) for these operations. In practice, implementations of asymmetric algorithms are orders of magnitude more expensive (in terms of resources) than their symmetric counterpart [32]. Yet, symmetric algorithms require to securely exchange the secret key as a first step. An (offline) asymmetric encryption can be used for that purpose.

Most of the discussion and results reported in this thesis are based on the AES, that is an *iterated block cipher*. In general, block ciphers are symmetric encryption primitives that, as their name indicates, operate on n -bit “blocks” of data (typically 64, 128, or 256-bit wide). As represented on Figure 2.1, the *plaintext* p is transformed into a *ciphertext* c under the action of the secret key κ . The plaintext can be recovered from the ciphertext through the decryption process, but only if the same key is used. In security definitions [48], the block cipher can be seen as a keyed permutation F_κ that is a bijective function mapping n -bit strings to n -bit strings. The goal is to make F_κ indistinguishable from a pseudorandom permutation. That is, the adversary should not be able to guess which of the permutations was used to generate the output he has access to. The encryption is written as $c = F_\kappa(p)$ and the decryption as $p = F_\kappa^{-1}(c)$. It is designed such that computing F or F^{-1} with the knowledge of the key is easy (polynomial-time), while retrieving the key from p or c (or both simultaneously) is not.

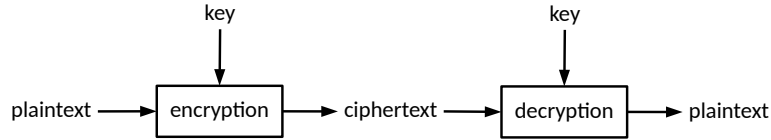


Figure 2.1: Symmetric encryption/decryption process

In order to allow efficient implementations on a wide range of platforms, block ciphers are usually made of operations that are easy to compute independently. Iterated block ciphers come from the observation that a repeated application of a transformation that is weak by itself can lead to a strong cipher [92]. In general, the transformation is called a *round* and is built with different components. Every round transformation inserts a round key that is derived from the master key κ by means of a *key scheduling* algorithm. We call *state* the intermediate values in the round transformation. The two most common round structures are the Feistel network and Substitution-Permutation Network (SPN). The structure of the AES is an SPN. Hence it is the focus of the rest of the section. Moreover it was designed with the wide trail design strategy [27]. In this context, the SPN-based round transformation is made of three layers:

- The *key addition* is the operation that inserts the round key in the encryption/decryption process, by mixing it with the *state*.
- The *substitution* layer consists in a complex non-linear transformation on bundle of bits (neighboring bits) of the state. Since these operations involve a few bits at a time, it is usually implemented with look-up tables, called S-boxes. Yet, some hardware or bitslice implementations may require a combinatorial representation [56].
- The *diffusion* layer is used in order to propagate the information through the state bits. It is defined such that neighboring bits in a round are not neighbors in the next one. The substitution layer operates on bundles, hence there is no interaction between them, i.e. no diffusion. The diffusion layer provides the required inter-bundle interaction for thwarting cryptanalysis attacks. This layer is usually made of simple linear operations that can be efficiently implemented without look-up tables.

The purpose of the key schedule algorithm is to produce round keys from the master key that are mixed with the state in the key addition layer. It is mainly used to provide security against related keys

attacks [49]. Yet, the research in cryptography community is divided on its exact requirements. Some algorithms have no key schedule. The master key is then used at every round (e.g. LED [42]). Some algorithms have a key schedule as complex as the round transformation (e.g. KHAZAD [8]). The AES key schedule is a trade-off between these two extrema.

2.1.2 Advanced Encryption Standard

The AES Rijndael [26] was proposed by Joan Daemen and Vincent Rijmen as a candidate at the National Institute of Standards and Technology (NIST) contest for finding a new encryption standard. The purpose of this contest was to find a suitable substitute for the Data Encryption Standard (DES) that was becoming obsolete (56-bit keys). We have selected this algorithm to lead our experimentation for two main reasons. Since its official election in 2001, it has been the target of many attacks and other research. It is thus deeply studied and well-known, and comparison with other works is easy. Moreover, it is widely spread in the industry.

The AES comes in three different versions: all processing blocks of 128 bits but differing by their key sizes (namely 128, 192, and 256 bits) and the number of round iterations (10, 12, and 14 respectively). In the work reported in this thesis, we make use of the 128-bit key and 10-round version. The state and rounds keys are represented as a 4-by-4 matrices containing bytes in a column-major order. This describes a matrix arrangement for which consecutive elements in a column are contiguous in the memory storing these values. The AES round transformation (Figure 2.2) involves four different byte-oriented operations:

- *SubBytes* is the non-linear (substitution) layer of the AES. The S-box transformation is defined as an 8-bit multiplicative inverse in the Galois field $\text{GF}(2^8)$ combined with an affine transformation. It is applied on every state byte.
- *ShiftRows* is the first part of the diffusion layer. It operates on the rows of the state. A byte-wise circular shift on the left is applied on the last three rows, while the first row remains unchanged. The second, third and fourth rows are respectively shifted by one, two and three positions.
- *MixColumns* is the second part of the diffusion layer. It operates on the columns of the state and consists in applying a reversible linear transformation: a matrix multiplication of every column

with a fixed matrix with coefficients in $\text{GF}(2^8)$. Its combination with **ShiftRows** provides a good diffusion as reported in the wide trail design strategy [27].

- **AddRoundKey** is the key addition layer. The state and the round key are mixed with a bit-wise exclusive-or operation (XOR).

The AES execution (Figure 2.3) consists in a first (isolated) key addition between the plaintext and the first round key. Next, the round transformation is iterated ten times (**MixColumns** is skipped in the tenth iteration). The round keys are derived from the master key with the AES key schedule algorithm: the first round key is directly the master key while the others are generated with 32-bit word shifting, XOR operations, and S-boxes. A round constant is integrated in the key schedule in order to avoid slide attacks (and other structural attacks) that would take advantage of similarities between the rounds.

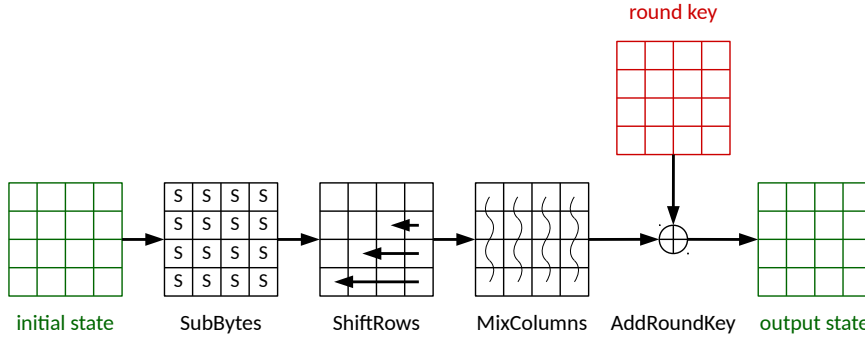


Figure 2.2: AES round operations

2.1.3 Software implementations

There are two main families of implementations (i.e. physical realizations), namely *hardware* and *software*. Generally, hardware implementations aim at providing high-end and optimized implementations with very high performances. They are built with directly assembling logics gates in order to implement every required mathematical function. It results in an integrated circuit that is optimized for the targeted application. However, the development process is long and expensive. For some applications, less performances may be sufficient. In this case, software implementations are often a suitable alternative. They make use of general purpose integrated circuits that are built with a set of

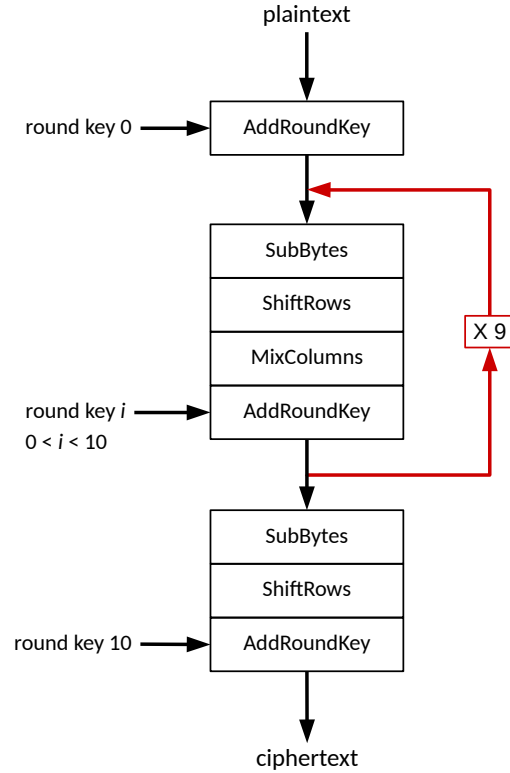


Figure 2.3: AES execution overview

pre-defined (hence not optimized) mathematical functions. These circuits are capable of executing a sequence of instructions defined by the user, called a *program*. Developing software (i.e. programming) usually goes faster than its hardware counterpart, but trades performances for a gain of development cost. This section describes *microcontrollers* that are a variety of these platforms.

General description. Microcontrollers are specially designed for embedded applications and integrate a whole electronic system. Their architecture (depicted on Figure 2.4) is usually composed of the following elements [44]:

- The *control unit* decodes the sequence of instructions defined in the program. It then drives and synchronizes all the components of the circuit to transfer the data and perform the requested operations.

- The *Arithmetic Logic Unit* (ALU) is the part that effectively computes the required mathematical operations. It is usually composed of a set of basic arithmetic operations. The control unit sets the one that is required by the instruction.
- The memory components come with different sizes and purposes. The *general purpose registers* are small and very fast memory elements used for storing the ALU operands and intermediate computation results. The *instruction memory* stores the program in the form of a *machine code* that can be interpreted by the control unit. The *data memory* stores the data to be processed and the computation results.
- The *peripherals* are other components that are driven by the control unit. They may be for internal usage (e.g. timer) or they may serve as the interface with the outside world (e.g. serial communication controller, analog-to-digital converter).
- The *system bus* is the inter-components communication part. This is basically parallel interconnections and is usually composed of three different parts: (1) The data bus is used for transferring the processed data. (2) The address bus carries the information about the recipient component. (3) The control bus contains all the control signals emitted by the control unit to synchronize the communicating components. The architectures are generally denominated by their data bus width (typically 8, 16, or 32-bit wide) which also limits the range of the data that can be processed.

The *Central Processing Unit* (CPU) is the core of the microcontroller. It includes the control unit, the general purpose registers, and the ALU. Microcontrollers may differ by their architecture, namely Von Neumann or Harvard [44, 74]. In the Von Neumann architecture, instruction and data memories share the same physical resources (memory component and bus). By contrast, the Harvard architecture uses two separate bus systems to carry the instructions to execute and the data to process. Most of today's microcontrollers are based on a modified Harvard architecture that allows fetching data from the instruction memory. Microcontrollers commonly use two kinds of memories, namely a Read-Only Memory (ROM) and a Random-Access Memory (RAM). As its name implies, the ROM can only be read by the CPU. Yet, some ROM technologies may be written but at the cost of a slow and difficult process, e.g. Flash or EEPROM. By contrast, RAM are in general faster to access and can be written and read by the CPU. However, they

are generally volatile and lose their content once the microcontroller is powered down, e.g. SRAM.

A typical instruction execution occurs in two cycles, namely fetch and execute cycles. During the former, the CPU reads the instruction to execute in the instruction memory. The latter carries out the operation indicated by the instruction. They do not necessarily take the same time, depending on the number of required instructions and their complexity. The Harvard architecture allows fetching the next instruction while the current one is executed.

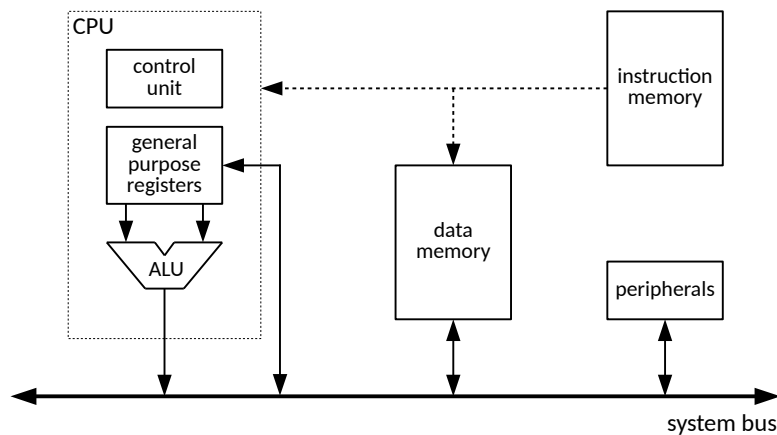


Figure 2.4: General architecture of a microcontroller

Programming languages. The program is the sequence of instructions defined by the user. The machine code is its representation and is stored in the instruction memory. Because directly manipulating operation codes and operands addresses is not an easy task, a *programming language* is used. Its purpose is to simplify the machine code development in order to make it more understandable by the developer. There are many different languages, corresponding to different levels of abstraction. They go from higher level ones such as C/C++, Python, or Java (and many others) to low level languages such as *assembly*. Higher level languages are generally compiled and/or interpreted. That is, they need an intermediate program to run in order to translate the high level instructions into machine code. In this case, predicting the resulting machine code is difficult. It may be significantly affected by the constraints set by the developer in charge of running the compiler/interpreter. By contrast, the assembly language is closer to the machine

code although it also requires a translation step. Yet, each developer instruction corresponds to exactly one subset of machine code instructions. It is generally used for bypassing compilers/interpreters artifacts (not always in control of the developer) and/or for optimizing software implementation performances. As developed in the next sections, the physical security of cryptographic implementations is highly dependent on the executed operations and processed data. This can be efficiently controlled in assembly implementations.

Performance metrics. Performances of microcontroller implementations can be evaluated considering different criteria [4]:

- The *code size* is the size (number of bytes) of the program stored in the instruction memory. Complex algorithmic operations generally imply more machine code instructions, hence a bigger code size. Some algorithms are designed such that this metric is mitigated, e.g. the NOEKEON block cipher [25]. Yet, this generally comes at the price of lower security margins.
- The *RAM use* is the space occupied by the intermediate values in the data memory. In the context of block ciphers, it can be due to the state and/or the expanded key (round keys) sizes.
- The *cycle count* is an image of the execution time. Integrated circuits operate at a given frequency that is dictated by the transition (edge) of a signal called the clock. A cycle corresponds to the period between two clock edges with the same sign (i.e. rising or falling edges). Every ALU operation requires a certain amount of clock cycles to be fully executed. The bigger is the number of operations the algorithm has to process, the larger is the cycle count. The actual execution time (in seconds) can be computed by dividing the cycle count by the clock frequency.
- The *combined metrics* aim to summarize the efficiency of an implementation. Multiplying (i) the code size and the cycle count, and (ii) the RAM use and the cycle count are good candidates in order to evaluate block cipher implementations on a fair basis.

2.1.4 Case-study: AES Furious

In this thesis, our experiments are based on an AES Furious implementation run by an 8-bit Atmel AVR (ATmega 644P) microcontroller at 20 MHz clock frequency.

The AES Furious [77] is a 128-bit AES software implementation written in an assembly language that targets the AVR microcontrollers family. It was developed with a small code size optimization goal which is mainly achieved by interleaving the `SubBytes` and `ShiftRows` layers, hence reducing the memory access. This implementation targets embedded devices and requires 1570 bytes of code size for both encryption and decryption, which are respectively computed within 2739 and 3579 clock cycles (resp. 137 and 179 μ s at 20 MHz, which corresponds to ~ 117 kB of encrypted data per second). The only source of RAM use is the key scheduling output, i.e. the round keys, that occupies 176 bytes.

The ATmega644P [3] is an 8-bit Atmel AVR microcontroller working at low clock frequencies up to a maximum operating frequency of 20 MHz (the one we choose for our setup). It is a modified Harvard architecture for which the instructions are stored in a separate physical memory with a different address space, but special instructions allow fetching data from that memory location. There are two ROM memories (Flash and EEPROM) of respectively 64k and 2048 bytes, and a RAM (SRAM) of 4kB.

2.2 Side-channel analysis

Block ciphers are designed such that they are resilient to “classical” cryptanalysis attacks. That is, they must be indistinguishable from a pseudorandom permutation, and it must not be computationally possible for a potential adversary to retrieve the secret key by only having access to the plaintext and/or the ciphertext. This relies on the relatively strong assumption that he has no physical access to the device. Yet, it is not always verified in practice, especially in the context of embedded devices. In some cases, the adversary has the possibility to affect and monitor the physical features of the device during the execution of the encryption algorithm. Exploiting these unintended side-effects is called *physical cryptanalysis*. Since it was first described in the 90’s [55], it has proved to be a real threat against embedded devices. Physical attacks are usually classified according two orthogonal axes [64]: *invasive* vs. *non-invasive* and *active* vs. *passive*.

- In invasive attacks, the device is physically tampered by the adversary in order to get access (or at least facilitate the access) to the information about the processed data. It can for instance be done by removing or altering the package of the chip. Non-invasive attack only exploit physical leakages of the device that can be

measured externally. That is, conversely to invasive attacks, no damage is involved, hence making them harder to detect.

- Active attacks imply modifications of the chip behaviour in order to extract information about the processed data. It can be for instance, switching the values contained in the memory thanks to a laser beam, modifying the power supply voltage, or generating glitches on the clock signal. The goal is in general to induce faults in the algorithm execution [7]. By contrast, passive attacks exploit physical leakages recorded without altering the behaviour of the device.

In this thesis, we focus on *side-channel power analysis attacks* that are non-invasive and passive physical attacks. They exploit *physical leakages* that are measurements of the instantaneous power consumption of the device during the algorithm execution. This section first introduces this kind of physical leakages and their origin. We then provide a general description of the Differential Power Analysis (DPA) that exploits the data dependency in leakages. Next, we briefly introduce the existing countermeasures developed in order to thwart these attacks. More attention is given to a precomputed masking instantiation. Finally, we introduce the Eurocrypt 2009 framework that defines useful metrics for an evaluator's analysis.

2.2.1 Power traces

In general, when an algorithm implementation is processing data, some of its physical features are reflecting its activity. These are physical leakages. In the side-channel research community, the most popular ones are timing [54], power consumption [55], and electromagnetic radiations [87]. This thesis exclusively focuses on power consumption leakages.

Side-channel attacks are based on the observation that physical leakages vary with the executed operations and the processed data. In the case of cryptographic algorithms, it implies that leakages possibly contain some information about the manipulated data, including data dependent on the key. The power consumption of integrated circuits carries information about the processing activity because of their construction. Let us represent these circuits as a set of logic cells. The total power consumption, denoted as P_{total} , essentially depends on the number of logic cells. It can be divided into two parts: the static and dynamic power consumptions. The former is due to leakage currents drawn by the idle logic cells. The latter is proportional to the switching activity,

i.e. the bit transitions $0 \rightarrow 1$ and $1 \rightarrow 0$. Four contributions to the total power consumption based on these sources can be identified [64]:

1. The executed operations mainly differ by the amounts of working logic cells. Hence, they impact both static (due to the number that stays idle) and dynamic components (due to those that are effectively working). Their contribution to the total consumption is referred to as $P_{operation}$.
2. Processing the sensitive data (containing information about the key) directly affect the number of bit transitions, hence impact the dynamic part. Its contribution is referred to as P_{data} .
3. Sometimes, other (irrelevant) data are processed in parallel. This creates additional switching activity and causes *algorithmic noise*. Its impact is denoted as P_{algo_noise} .
4. The last (but not the least) contribution is the *electronic noise*. It essentially includes all the physical noise (e.g. power supply noise, clock noise, radiations) and the measurement noise (e.g. quantization). It is next denoted as P_{elec_noise} .

A *power trace* is a measure of the total power consumption P_{total} of a cryptographic device which is a function P of the aforementioned contributions (they are all functions of the time). For simplicity, the function P can be modelled as a sum:

$$\begin{aligned} P_{total} &= P(P_{operation}, P_{data}, P_{algo_noise}, P_{elec_noise}) \\ &= P_{operation} + P_{data} + P_{algo_noise} + P_{elec_noise} \end{aligned}$$

An example of a power trace of a complete AES execution on an AVR microcontroller (Section 2.1.4) is presented on Figure 2.5 (smoothed for legibility purpose). The algorithm is easily recognized thanks to the round pattern that is reproduced ten times (the last is shorter due to MixColumns absence). The eleventh peak corresponds to the extra key addition. The relative difference of peaks amplitude results from the different values of the state at every round iteration.

Attacks that exploit operation and data variations on a single power trace are usually referred to as Simple Power Analysis (SPA) [55]. They are particularly effective if a sequence of operations depends on the sensitive data. For instance, some modular exponentiation methods may require an extra multiplication if the exponent value (the key) equals 1. Exploiting the presence (or not) of the multiplication in the leakage trace directly returns the key bits. Block ciphers are generally safe

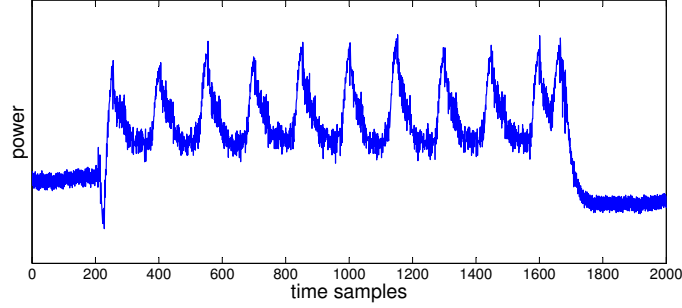


Figure 2.5: AES leakage traces

against SPA attacks since they consist in executing the same sequence of operations, no matter what is the key value. Yet, they are often an interesting preliminary step for extracting relevant features of the algorithm, e.g. the position of the rounds. Our focus is on Differential Power Analysis (DPA) [55] that only exploits the data dependency on multiple leakage traces.

2.2.2 Standard DPA

Block ciphers are usually designed such that small size blocks (typically 8-bit) can be processed independently, providing efficient implementations on all kinds of platforms. Even though they can have different structures, the beginning of the algorithm is often similar. Figure 2.6 illustrates a typical 8-bit key addition and S-box layer in the first round of the AES. This is a usual target for side-channel attacks since these operations only involve a known plaintext byte x and the secret subkey s (i.e. key byte). In this case, the adversary can exploit the information leaked about either of the operations results, respectively denoted as y and z . He follows a *divide-and-conquer* strategy. During the “divide” step, all subkeys are attacked and recovered separately. In the “conquer” step, they are assembled in order to recover the whole secret key κ .

The thesis focuses on DPA attacks that can generally be described as illustrated on Figure 2.7 [10, 65]. Let us consider that the adversary targets the output z of the S-box. For multiple plaintexts x_i and a fixed subkey s , he has leakage samples l_z^i that are generated by a *leakage function* depending on the intermediate values z_i . The exact behaviour of this function depends on physical features on the processing devices and is considered unknown to the adversary. The DPA attack follows

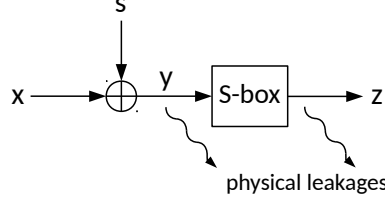


Figure 2.6: Leaking S-box of the first AES round

three main steps:

1. From the plaintext x_i and different subkey candidates s^* , the adversary *predicts* the intermediate values of the implementation. Under Kerckhoff's principle [51] which states that a cryptosystem should be secure even if everything, except the key, is public knowledge, we consider that the adversary knows the target algorithm details. In this case, as he targets the outputs z_i of the S-box, the prediction consists in computing $z_i^* = S(x_i \oplus s^*)$.
2. For each of the predicted values, the adversary then *models* the leakages as $\text{model}(z_i^*)$. For that purpose, he makes use of a *leakage model* that aims to emulate the leakage function.
3. For each of the subkey candidates s^* , the approximated values $\text{model}(z_i^*)$ are compared with the actual leakages l_z^i . For that purpose, the adversary makes use of a *distinguisher* that returns the similarity between the two sets of values.

An attack succeeds when the subkey candidate returning the highest similarity corresponds to the actual subkey. It can easily be verified by performing the encryption of a plaintext with the recovered key and by comparing the output ciphertext with the original one. If it fails, it can either be caused by a wrong leakage model, or because the targeted implementation is secure (probably due to noise or countermeasures). The literature provides lots of solutions for the modelling step and lots of statistical tools that can be used as distinguisher. We list those we consider for our experiments in a next section.

DPA attacks are usually classified according to three orthogonal axes: *non-profiled* vs. *profiled*, *univariate* vs. *multivariate*, and the *order*.

- In a non-profiled attack scenario, the adversary only has access to the *attack set* of traces. That is, the plaintext is known/controlled

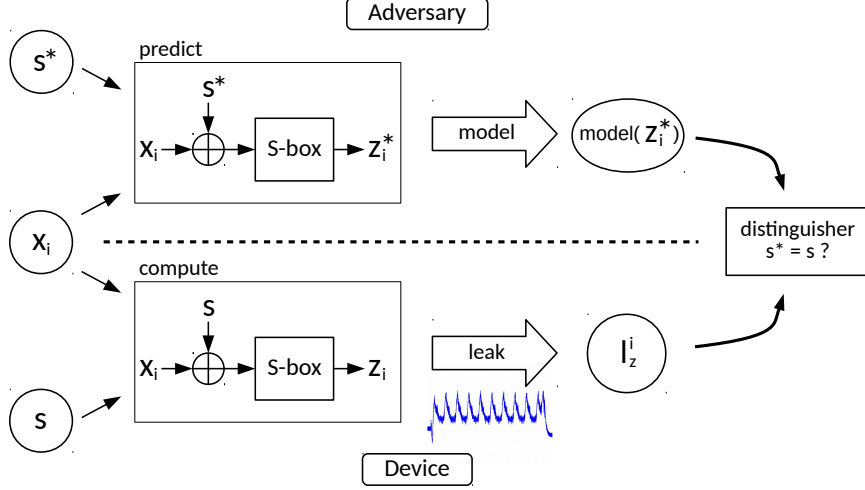


Figure 2.7: DPA attack

while the key remains secret. In order to model the leakage function, he makes (strong) assumptions (e.g. Hamming weight [23]) that exploit his knowledge of the device. In a profiled attack setting, the adversary has access to the same implementation of the targeted algorithm with the possibility to control the plaintext as well as the key. It allows him to build a *profiling set* of traces in order to train his leakage model on actual observations. This training step (denoted as profiling) allows relaxing assumptions on the leakage model even though a few (lighter) ones remain to be made (e.g. Gaussian noise distributions [16]). The work reported in this thesis mainly focuses on profiled side-channel analysis.

- In general, univariate (one) or multivariate (at least two) denotes the number of dimensions that are simultaneously observed within a given statistic. In the context of DPA, this mostly depends on the distinguisher that is used, i.e. if it can handle more than one dimension at the same time. The dimensions relate to the time samples that may contain information about the same or different intermediate values. However, they may be combined or compressed with *dimensionality reduction* techniques, e.g. Principal Component Analysis (PCA) [2] or Linear Discriminant Analysis (LDA) [103]. As their name implies, they are used to reduce the number of dimensions to handle. They can be used for instance in order to evaluate multiple time samples within a univariate statis-

tic.

- The order of an attack refers to the statistical moment that needs to be observed for the attack to succeed. The power traces described in Section 2.2.1 are *first-order* leakages if the information about the sensitive data is in the mean. An attack that would need to exploit a moment of a higher order d (e.g. variance, skewness, kurtosis) is denoted as d^{th} -order.

2.2.3 Countermeasures

Side-channel attacks have appeared to be a real threat against cryptographic devices. Since they were discovered, lots of efforts have been made in order to mitigate their impact by the mean of countermeasures. They aim to remove the sensitive data dependency in physical leakages and can be integrated at three different levels: *hardware*, *algorithmic*, and *protocol* levels. In general, they mitigate the data dependency but do not completely remove it. Defining and implementing a countermeasure is typically a matter of trade-off between performances and security. In industrial applications, several countermeasures are usually combined in order to increase the security against side-channel analysis. Attacking multiple countermeasures at the same time remains a real challenge [94]. This section briefly introduces the different kinds of countermeasures provided in the literature. More details about a pre-computed masking scheme is given next. It is used as a case-study in the point-of-interest detection work that is reported in this thesis.

General description

Hardware countermeasures work at the logic gate level and generally make use of a dedicated logic style designed to reduce the data dependency. One of the main advantages of gate-level countermeasures is that they do not depend on the targeted algorithm. Hence, they can be easily applied to any other cryptographic primitive. Literature proposes solutions such as Dual-Rail Precharge (PRD) logic style, also called Dynamic and Differential Logic (DDL) [109, 110]. The basic idea behind this solution is that every manipulated bit is encoded on two wires, one with its value, the other with its negative: $0 \rightarrow (0, 1)$ and $1 \rightarrow (1, 0)$. The consumption is then expected to remain constant, whatever the processed value. In practice, perfectly balancing the two wires is not feasible. Hence some information goes on leaking.

Algorithmic countermeasures work at the implementation level (in the

behavioural description, i.e. in the programming process in the case of microcontrollers). Using this type of countermeasure is usually at the cost of decreased performances and a higher algorithmic complexity. Moreover, they are generally specific to a targeted algorithm, thus they cannot be used for every cryptographic primitive. This kind of countermeasure is generally classified in two main families, namely *hiding* and *masking* countermeasures.

- The hiding countermeasures aim to, as their name implies, hide the information in the leakages. A standard side-channel attack relies on the assumption that the same data is always processed at the same time instant. Hiding countermeasures usually consist in randomizing the time location of the processing of the targeted data. (1) The *shuffling* [45, 113] consists in randomizing the operation execution order. If the adversary is attacking an implementation protected with shuffling, he is still able to identify the operations, e.g. the S-box layer. However, he should not be able to determine their execution order. For instance, if an unprotected device processes the AES S-boxes as $(S_0, S_1, \dots, S_{15})$, a shuffled execution would be $(S_{i_0}, S_{i_1}, \dots, S_{i_{15}})$ where $(i_0, i_1, \dots, i_{15})$ is a random mapping and is regularly refreshed. (2) The *random delays* [111, 22] are sequences of dummy operations (or dummy execution of real algorithm operations) that are inserted at various times in the actual algorithm execution. The length of the delays is randomly set for every new encryption. This countermeasure has the effect of misaligning all the traces.
- The masking [41, 15] aims to directly randomize the processed data. This type of countermeasure is designed such that the sensitive data are never directly processed. Instead, the data are decomposed into $d + 1$ shares, for a d^{th} -order scheme [93]. For instance, in the case of a first-order masking scheme, the first share carries the sensitive data that are mixed with a *mask*, i.e. a random value that is generated internally, while the second share carries the mask. Both shares are processed independently and combined at the end of the algorithm in order to remove the mask. Masking schemes mainly differ by their combination function: (1) boolean masking uses a bitwise XOR operation (2) multiplicative masking uses a multiplication over $\text{GF}(2^8)$ (3) affine masking implements an affine mapping with look-up tables. In general, the higher is the order of the masking scheme, the safer is the implementation against side-channel attacks. In order to attack such protected devices, the adversary must consider the leakages of all the shares

at the same time. This generally requires high-order side-channel attacks. Yet, higher-order masked implementations usually come at the price of a huge overhead of resource consumption and execution time.

Protocol-level countermeasures are applied at a higher level than the encryption algorithm. For example, *re-keying* [69, 67] consists in regularly changing the secret key used by the device such that the same key is never used for more than a specified number of encryptions. It relies on the assumption that a side-channel attack requires a minimum number of traces to recover the key. Hence, changing the key within that number of encryptions should prevent attacks from succeeding. The main challenge is for the algorithm to generate the new keys. If not enough attention is paid, successive leakages from different encryptions can be used by the adversary to recover the master key [67]. In general, combining this of countermeasure with lower level ones ensures a better security against side-channel attacks. *Leakage resilience* cryptography formalizes this solution and provides proofs that cryptographic constructions such as Pseudo-Random Generators (PRG) [31, 108] and Pseudo-Random Functions (PRF) [28] are secure against side-channel attacks. PRGs and PRFs can be build from cryptographic primitives (e.g. block ciphers). This is however under strong assumptions, e.g. no more than a given number of bits of information can be extracted from the leakages of the primitives, which is hard to verify.

Pre-computed masking

In the reported work, we make use of a masked implementation of the AES S-box to validate our experiments. For this purpose, we consider the actual measurements of a first-order (boolean) masked AES S-box based on table lookups [82, 97]. For every pair of input/output masks (m, q) , it pre-computes an S-box S^* such that $S^*(x \oplus s \oplus m) = S(x \oplus s) \oplus q$ (illustrated on Figure 2.8), where S is the AES S-box. For our experiments, we focus on an 8-bit key addition and S-box layer in the first round of the AES, due to the large execution time overhead caused by the pre-computation part.

As described in Algorithm 1, the pre-computation and execution are in separate parts of the algorithm (and performed at separate times). The pre-computation consists in (1) exploring the 256 possible input values, (2) removing the input mask m , (3) applying the AES S-box S , and (4) protecting the output value with the output mask q . In order to execute the S-box S^* and keep the sensitive value protected,

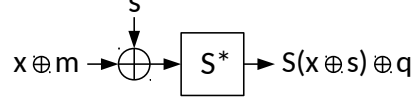


Figure 2.8: Pre-computed masked AES S-box

the input mask m is first applied to the (known) plaintext x . Then, the key addition is performed. The underlying idea is to avoid any direct manipulation of the sensitive values $y = (x \oplus s)$ or $z = S(y)$. The protected intermediate value $y_m = (y \oplus m)$ is substituted with the table S^* whose output $z_q = S(y) \oplus q$ is protected with a mask q . The masks m and q are uniformly distributed and drawn independently such that they are most likely different. When the table S^* is read, leakages depend on both the input and the output. For instance, if the masks were equal, a Hamming distance-like effect exposing the sensitive values would possibly appear: $\text{HD}(y \oplus m, z \oplus m) = \text{HW}(y \oplus z \oplus m \oplus m)$, where HD is the Hamming distance, and HW is the Hamming weight. The values m and q are generated on-chip and require to be regularly refreshed (the best is at every encryption). It is assumed they cannot be accessed by the adversary. With this kind of implementation and if he targets the S-box S output z , the adversary must simultaneously exploit leakages of the two shares, i.e. both the pre-computation (where q is processed) and the S-box S^* execution, where z_q is manipulated.

Algorithm 1 Protected S-box S^* pre-computation and execution

```

/* Pre-computation */
m = rand(0...255); q = rand(0...255);
for i = 0 → 255
  a = i ⊕ m;
  b = S(a);
  S*(i) = b ⊕ q;
end

/* Execution */
xm = x ⊕ m;
ym = xm ⊕ s;
zq = S*(ym);
  
```

2.2.4 Eurocrypt 2009 framework

The key recovering process of a side-channel attack is formally generalized in the framework reported at Eurocrypt 2009 [105]. It aims to tackle two very different questions that are (i) “how safe is the implementation?” and (ii) “how strong is the adversary?”. Authors propose a methodology for identifying and removing subjective parameters in order to evaluate implementations and attacks on a fair basis.

A general key recovery procedure is illustrated on Figure 2.9. We consider a device running the AES cryptographic primitive. The device leaks some information through a side-channel that is an unintended communication channel. The information is carried by a physical media (e.g. power consumption, electromagnetic radiation) that is physically observable. The leakage function is an abstraction that models the side-channel as well as the measurement setup used to monitor the physical observables. Its output is the physical leakage. The side-channel adversary is defined as an algorithm that queries the implementation in order to get physical leakages of the algorithm execution in addition to the classical access. The adversary’s goal is to recover the key within certain computational bounds and capabilities.

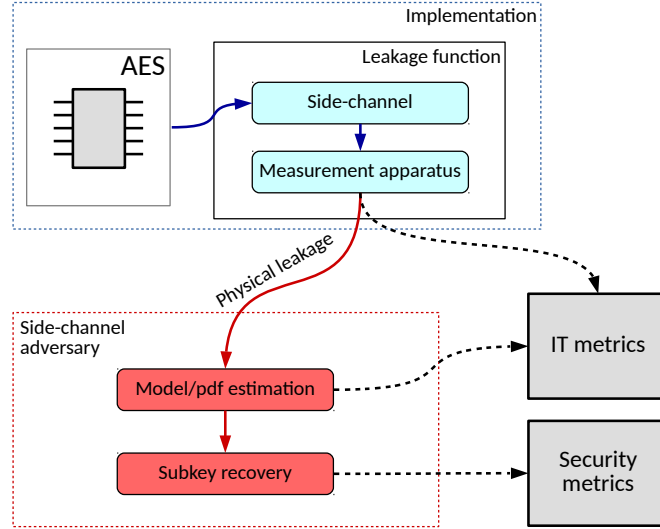


Figure 2.9: Side-channel key recovery

Given this representation, two aspects have to be considered. First, it is important to note that actual implementation leaks information

that is independent of the adversary exploiting it. *Information Theoretic (IT) metrics* are used in order to give a sound answer to the first question, i.e. regarding the implementation quality. The metric that is considered in this thesis is the *mutual information* (Section 2.3.3) that is based on Shannon’s conditional entropy. Second, the adversary exploits these leakages whose interpretation depends on his assumptions. *Security metrics* measure the extent to which this exploitation effectively uses the available information. Therefore it aims to answer the second question that involves gauging the adversary’s strength. For this purpose, we make use of the *success rate*, or more precisely the first-order success rate. It is defined as the probability that the actual secret key κ is ranked at the first position in the most probable keys at the end of the side-channel attack. The success rate is returned for a given number of attack traces and a given model.

2.3 Instantiation

All the work reported in this thesis is related to DPA attacks on an 8-bit AES implementation. This section introduces our measurement setup and all the statistics from the literature on which our experiments are based.

2.3.1 Measurement setup

Our experiments are based on measurements of an AES implementation run by an 8-bit Atmel AVR (ATmega644P) microcontroller at a 20 MHz clock frequency. We monitored the voltage variations across a 22 Ω resistor introduced in the supply circuit of our target chip. Acquisitions were performed using a Lecroy HRO66ZI oscilloscope running at 200 MHz and providing 8-bit samples. For concreteness, most of our evaluations focused on the leakage of the first AES master key byte (but would apply identically to any other enumerable target). For illustration, leakage traces were produced according to the following procedure. Let x and s be our target input plaintext byte and subkey, and $y = x \oplus s$ denote a key addition. For each of the 256 values of y , we generated 1000 unprotected encryption traces (resp. 500 for masked traces), where the rest of the plaintext and key was random, i.e. we generated 256 000 (resp. 128 000) traces in total, with plaintexts of the shape $p = x || r_1 || \dots || r_{15}$, keys of the shape $\kappa = s || r_{16} || \dots || r_{30}$, and the r_i ’s denoting uniformly random bytes. In case of masked implementations, additional uniform randomness was used to generate the shares (Section 2.2.3). In order to reduce the memory cost of our evaluations, we only stored the leakages

corresponding to the first 2 AES rounds in the unprotected case (as the dependencies in our target byte $y = x \oplus s$ typically vanish after the first round, because of the strong diffusion properties of the AES). As for the protected case, we only considered a single S-box, for which the precomputation of a masked table already implies large traces with $N_s = 30,000$ time samples (vs. $N_s = 1500$ for the unprotected one). As will be clear further, these sets of measurements were large enough to emphasize the interest of our evaluations goals. In the following, we denote the 1000 (resp. 500) encryption traces obtained from a plaintext p including the target byte x under a key κ including the subkey s as: $\text{AES}_{\kappa_s}(p_x) \rightsquigarrow l_y^i$, with $y = x \oplus s$ and $i \in [1; 1000]$ (resp. $i \in [1; 500]$). Whenever accessing the points of these traces, we use the notation $l_y^i(\tau)$, with $\tau \in [0; N_s - 1]$. These subscripts and indexes will be omitted when not necessary. We denote a set of traces, for a given kind of inputs (typically for profiling or attacking), as $\mathcal{L}$. Note that the sets of traces that are subsequently used for validating our experiments may slightly differ from this general description. In this case, the new set and its differences are introduced at the beginning of the chapter.

2.3.2 PDF estimation methods

Side-channel attacks such as the standard DPA described in Section 2.2.2 require a leakage model. It can either be non-profiled (based on knowledge of the device), or profiled (estimated from actual measurements). In the latter case, such models generally correspond to estimations of the leakage Probability Density Function (PDF – possibly simplified to certain statistical moments). In the following, we will consider two important PDF estimation techniques for this purpose.

Gaussian templates

The Template Attack (TA) in [16] approximates the leakages using a set of normal distributions. It assumes that each intermediate computation generates Gaussian-distributed samples. In our typical scenario where the targets follow a key addition, we consequently use: $\hat{\text{Pr}}_{\text{model}}[l_y|s, x] \approx \hat{\text{Pr}}_{\text{model}}[l_y|s \oplus x] \sim \mathcal{N}(\mu_y, \sigma_y^2)$, where the “hat” notation is used to denote the estimation of a statistic. This approach requires estimating the sample means and variances for each value $y = x \oplus s$ (and mean vectors / covariance matrices in case of multivariate attacks). We denote the construction of such a model with $\hat{\text{Pr}}_{\text{model}}^{\text{ta}} \leftarrow \mathcal{L}_Y^p$, where $\mathcal{L}_Y^p$ is a set of N_p traces used for profiling.

Regression-based models

To reduce the data complexity of the profiling, an alternative approach proposed by Schindler et al. is to exploit Linear Regression (LR) [96]. In this case, a stochastic model $\hat{\theta}(y)$ is used to approximate the leakage function and built from a linear basis $\mathbf{g}(y) = \{\mathbf{g}_0(y), \dots, \mathbf{g}_{B-1}(y)\}$ chosen by the adversary/evaluator (usually $\mathbf{g}_i(y)$ are monomials in the bits of y). Evaluating $\theta(y)$ boils down to estimating the coefficients α_i such that the vector $\hat{\theta}(y) = \sum_i \alpha_i \mathbf{g}_i(y)$ is a least-square approximation of the measured leakages L_y . In general, an interesting feature of such models is that they allow trading profiling efforts for online attack complexity, by adapting the basis $\mathbf{g}(y)$. That is, a simpler model with fewer parameters will converge for smaller values of N_p , but a more complex model can potentially approximate the real leakage function more accurately. Compared to Gaussian templates, another feature of this approach is that only a single variance (or covariance matrix) is estimated for capturing the noise (i.e. it relies on an assumption of homoscedastic errors). Again, we denote the constructions of such a model with $\hat{P}_{\text{model}}^{\text{lr}} \leftarrow \mathcal{L}_Y^p$.

Histograms and kernels

The previous estimation methods make the assumption that the non-deterministic part of the leakage behaves according to a normal distribution. This may not always be correct, in which case other techniques need to be used. For illustration, we considered two non-parametric solutions for density estimation, namely histograms and kernels. These allow the non-deterministic part of the leakage to be finely characterized. First, *histogram* estimation performs a partition of the samples by grouping them into bins. More precisely, each bin contains the samples of which the value falls into a certain range. The respective ranges of the bins have equal width and form a partition of the range between the extreme values of the samples. Using this method, one approximates a probability by dividing the number of samples that fall within a bin by the total number of samples. The optimal choice for the bin width h is an issue in statistical theory, as different bin sizes can have great impact on the estimation. In our case, we were able to tune this bin width according to the sensitivity of the oscilloscope. Second, *kernel* density estimation is a generalization of histograms. Instead of bundling samples together in bins, it adds (for each observed sample) a small kernel centered on the value of the leakage to the estimated PDF. The resulting estimation is a sum of small “bumps” that is much smoother than the corresponding histogram, which can be desirable when estimating

a continuous distribution. In such cases, kernels usually provide faster convergence towards the true distribution. Similarly to histograms, the most important parameter is the bandwidth h . In our case, we used the modified rule of thumb estimator in [98].

2.3.3 Metrics

In this section, we present useful statistics that were introduced in previous works on side-channel attacks and countermeasures. Most of these metrics were originally introduced as distinguishers for standard DPA attacks. We rely on them for extended evaluations as introduced in the next chapters.

Difference of means

The first distinguisher proposed for DPA attacks is the difference of means [55]. This technique is non-profiled and is based on partitioning the set of traces into two different classes. Traces are sorted according to the value of an arbitrary selected bit of the adversary's predictions z^* (Section 2.2.2). The two resulting subsets are denoted as $\mathcal{L}_0$ and $\mathcal{L}_1$ and the metric is computed as:

$$D^*(\tau) = | \hat{E}_i(L_0^i(\tau)) - \hat{E}_i(L_1^i(\tau)) |$$

where $\hat{E}$ is the sample mean operator. The correct subkey candidate s^* is the one that maximizes the difference between these mean values.

Correlation coefficient

In view of the popularity of the Correlation Power Analysis (CPA) distinguisher in the literature [13], the second metric to introduce is Pearson's correlation coefficient. In the non-profiled setting, an a priori (e.g. Hamming weight) model is used to compute the metric. The evaluator then estimates the correlation between his measured leakages and the modeled leakages of a target intermediate value. In our AES example, it would lead to $\hat{\rho}(L_Y(\tau), \text{model}_{\text{cpa}}(Y))$. In practice, this estimation is performed by sampling (i.e. measuring) N_t test traces from the leakage distribution (we denote the set of these N_t test traces as $\mathcal{L}_Y^t$). Next, and in order to avoid possible biases due to an incorrect a priori choice of leakage model, a natural solution is to extend the previous proposal to the profiled setting. In this case, the evaluator will start by estimating a model from N_p profiling traces: $\hat{\text{model}}_{\text{cpa}} \leftarrow \mathcal{L}_Y^p$ (with $\mathcal{L}_Y^p \perp \mathcal{L}_Y^t$). In practice, $\hat{\text{model}}_{\text{cpa}}$ can be seen as a simplification of the previous

Gaussian templates (Section 2.3.2) that only include estimates for the first-order moments of the leakages. That is, for any time sample τ , we have $\hat{\text{model}}_{\text{cpa}}(y) = \hat{m}_y^1(\tau) = \hat{\mathbf{E}}_i(L_y^i(\tau))$, with $\hat{m}_y^1$ a first-order moment and $\hat{\mathbf{E}}$ the sample mean operator.

Signal-to-Noise Ratio

Introduced by Mangard in the side-channel analysis context [62], the Signal-to-Noise Ratio (SNR) of the measurements is defined as:

$$\text{SNR} = \frac{\hat{\text{var}}_y(\hat{\mathbf{E}}_i(L_y^i))}{\hat{\mathbf{E}}_y(\hat{\text{var}}_i(L_y^i))},$$

where $\hat{\mathbf{E}}$ and $\hat{\text{var}}$ denote the sample mean and variance of the leakage variable that are estimated from the N_t traces in $\mathcal{L}_Y^t$ (like the correlation coefficient).

Mutual and perceived information

In theory, the worst-case security level (i.e. the maximum amount of available information) of an implementation can be measured with a Mutual Information (MI) metric. Taking advantage of the notations in Section 2.3.1 and considering the standard case where a key byte S is targeted, it amounts to estimating:

$$\begin{aligned} \hat{\text{MI}}(S; X, L) = \\ \mathbb{H}[S] + \sum_{s \in \mathcal{S}} \Pr[s] \sum_{x \in \mathcal{X}} \Pr[x] \sum_{l_y^i \in \mathcal{L}_t} \Pr_{\text{chip}}[l_y^i | s, x] \cdot \log_2 \hat{\Pr}_{\text{chip}}[s | x, l_y^i] \end{aligned}$$

When summing up all s and x values, and a sufficiently large number of leakages, the estimation tends to the correct MI. Yet, as mentioned in Section 2.2.2, the leakage function is generally unknown to the evaluator, and so is the chip distribution $\Pr_{\text{chip}}[l_y^i | s, x]$. Then, in practice, the best that we can hope is to compute the following Perceived Information (PI):

$$\begin{aligned} \hat{\text{PI}}(S; X, L) = \\ \mathbb{H}[S] + \sum_{s \in \mathcal{S}} \Pr[s] \sum_{x \in \mathcal{X}} \Pr[x] \sum_{l_y^i \in \mathcal{L}_t} \Pr_{\text{chip}}[l_y^i | s, x] \cdot \log_2 \hat{\Pr}_{\text{model}}[s | x, l_y^i] \end{aligned}$$

where $\hat{\Pr}_{\text{model}} \leftarrow \mathcal{L}_Y^p$ is typically obtained using the previous Gaussian templates or LR-based models (Section 2.3.2). Under the assumption that the model is properly estimated, it is shown in [65] that the CPA

and PI metrics are essentially equivalent in the context of standard univariate side-channel attacks (i.e. exploiting a single leakage point $l_y^i(\tau)$ at a time). By contrast, only the PI naturally extends to multivariate attacks. It can be interpreted as the amount of information leakage that will be exploited by an adversary using an estimated model. So just as the MI is a good predictor for the success rate of an ideal TA exploiting the perfect model Pr_{chip} , the PI is a good predictor for the success rate of an actual TA exploiting the “best available” model $\hat{\text{Pr}}_{\text{model}}$ obtained thanks to profiling.

Moments-correlating DPA

Eventually, and in order to extend the CPA distinguisher to higher-order moments, the Moments-Correlating Profiled DPA (MCP-DPA) was introduced in [70]. It features essentially the same steps as a profiled CPA. The only difference is that the adversary first estimates d^{th} -order statistical moments with his profiling traces, and then uses $\hat{\text{model}}_{\text{mcp-dpa}}^d(y) = \hat{m}_y^d(\tau)$, with $\hat{m}_y^d$ a d^{th} -order moment. For concreteness, in our experiments, we consider d ’s up to four (i.e. the sample mean for $d = 1$, variance for $d = 2$, skewness for $d = 3$ and kurtosis for $d = 4$). This allows us to discuss the relevant case-study of a masked implementation with two shares. Yet, the tool naturally extends to any d . One useful feature of this distinguisher is that it embeds the same “metric” intuition as CPA: the higher the correlation estimated with MCP-DPA, the more efficient the corresponding attack exploiting a moment of given order.

Regarding our work on the points-of-interest detection, we will be particularly interested in the Moments against Moments Profiled Correlation (MMPC) criteria:

$$\text{MMPC}(\tau) = \hat{\rho}(\hat{\mathbf{m}}_y^d(\tau), \tilde{\mathbf{m}}_y^d(\tau)),$$

where $\tilde{\mathbf{m}}_y^d(\tau)$ are another vector of moments, estimated with the test traces. As detailed in [70], MCP-DPA is able to capture information in any statistical moment, while enjoying the implementation efficiency of CPA (which is highly beneficial in our context where it will be intensively used).

2.3.4 Estimating a metric with cross-validation

In a profiled side-channel analysis setting, estimating a metric α from a leaking implementation holds in two steps. First, a model has to be

estimated from a set of profiling traces $\mathcal{L}_p$: $\text{model} \leftarrow \mathcal{L}_p$. Second, a set of test traces $\mathcal{L}_t$ (following the true distribution Pr_{chip}) is used to estimate the metric: $\hat{\alpha} \leftarrow (\mathcal{L}_t, \text{model})$. As a result, two main types of errors can arise. First, the number of traces in the profiling set may be too low to estimate the model accurately (which corresponds to estimation errors). Second, the model may not be able to accurately predict the distribution of samples in the test set, even after intensive profiling (which then corresponds to assumption errors).

In order to verify that estimations in a security evaluation are sufficiently accurate, the solution used in the next chapters is to exploit cross-validation. In general, this technique allows gauging how well a predictive (here leakage) model performs in practice. For k -fold cross-validations, the set of evaluation traces $\mathcal{L}$ is first split into k (non overlapping) sets $\mathcal{L}^{(i)}$ of approximately the same size. Let us define the profiling sets $\mathcal{L}_p^{(j)} = \bigcup_{i \neq j} \mathcal{L}^{(i)}$ and the test sets $\mathcal{L}_t^{(j)} = \mathcal{L} \setminus \mathcal{L}_p^{(j)}$. The sample metric is then repeatedly computed k times for $1 \leq j \leq k$ as follows. First, we build a model from a profiling set: $\text{model}^{(j)} \leftarrow \mathcal{L}_p^{(j)}$. Then we estimate the metric with the associated test set $\hat{\alpha}^{(j)} \leftarrow (\mathcal{L}_t^{(j)}, \text{model}^{(j)})$. Cross-validation protects evaluators from obtaining too optimistic sample metric values due to over-fitting, since the test computations are always performed with an independent data set. Finally, the k outputs can be averaged in order to get an unbiased metric estimate, and their spread characterizes the result accuracy.

2.4 SCA workflow

An intuitive description of a side-channel attack is given in Section 2.2.2 and a formalized framework for their evaluation in Section 2.2.4. Yet, these representations do not take into account some preliminary procedures that are generally required in practice. Figure 2.10 informally illustrates the complete workflow of a practical side-channel analysis. It can be decomposed into five steps, having different purposes and requiring different skills and tools:

1. **Measure:** this is the process of recording the physical leakages with measurement equipments. It is the interface between the analog real world and the digital evaluator's work environment. This step is made thanks to analog-to-digital apparatus (e.g. oscilloscopes, see Section 2.3.1). The parameters of the measurement process (e.g. sampling frequency, trigger event, requests order) may significantly affect the attack outcome.

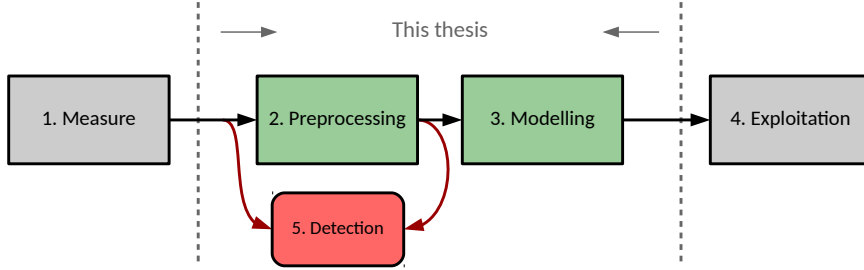


Figure 2.10: Steps of a complete side-channel analysis

2. **Preprocessing:** it consists in signal processing techniques that are used in order to increase the attack efficiency. Sometimes, they are even necessary to make it succeed. They are for instance methods for enhancing the measurement quality (e.g. filtering [71]), for detecting the (hidden) time samples that contain useful information [91], i.e. the points-of-interest, or for combining that information captured within different POIs [2, 103] (e.g. lower dimensional subspace projection).
3. **Modelling:** it is the step that aims to predict the leakage function outcome by approximating its behaviour with a model. As mentioned in Section 2.2.2, it can be either entirely based on assumptions (e.g. Hamming weight, Hamming distance), or estimated (i.e. profiled) with actual observations of the leakages (i.e. measurements) and with less (or lighter) assumptions. In our experiments, we focus on profiled settings (Section 2.3.2). During the profiling, the evaluator is able to manipulate the plaintext and the key such that the targeted intermediate values are processed. The leakage function is then estimated thanks to, for instance, PDF estimation methods. IT metrics (described in Section 2.2.4) are computed at this level in order to determine the amount of meaningful information that is captured by the model.
4. **Exploitation:** this is the process of exploiting the time samples in order to recover the secret key. The exploitation outcome is significantly affected by the quality of the previous steps realization. In general, it follows the same procedure as described in Section 2.2.2 (minus the modelling step that is here considered independently) and makes use of distinguishers such as introduced in Section 2.3.3. Security metrics (described in Section 2.2.4), that determine the

quality of attack, are computed at this level.

5. **Detection:** like the preprocessing, the detection is a preliminary step, yet with a different (orthogonal) goal. It aims to detect whether leakages of a given order are effectively present in leakage traces or not. Therefore, the result returned by this process may not indicate the points-of-interest, i.e. the points that can effectively be used for an attack. Moreover, it may not reflect the amount of information that is actually contained in the leakage. It mainly depends on the statistical metric used by the evaluator for this purpose.

In this thesis, we focus on the preprocessing, modelling and detection steps of a practical side-channel analysis. The first part of our work investigates points-of-interest detection and the relation with leakage detection. For that purpose, we make the comparison with the widely spread Cryptography Research (CRI)’s non specific (fixed vs. random) t-test that is introduced in the next sub-section. The second part is focused on evaluating the quality of models that are built in a profiled setting. This latter part of the work aims to tackle the question “how can we make sure that our model reflects the reality?”.

2.4.1 Fixed vs. random leakage detection test

Introduced by Cryptography Research (the company created by Paul Kocher), the fixed vs. random t-test [38] takes advantage of leakage obtained from two different sets of inputs, namely fixed and random. The first corresponds to a fixed plaintext and key, while the other corresponds to random plaintexts and a fixed key. They are next denoted as $\mathcal{L}_f$ and $\mathcal{L}_r$ respectively. This test is inspired from the difference of mean distinguisher (Section 2.3.3) and essentially works by comparing the leakages corresponding to these two sets. For this purpose, and for each sample, one simply has to estimate and compare two mean values. The first one, denoted as $\hat{\mu}_f(\tau)$, corresponds to the samples in the fixed set of traces $\mathcal{L}_f$. The second one, denoted as $\hat{\mu}_r(\tau)$, corresponds to the samples in the random set of traces $\mathcal{L}_r$. Intuitively, being able to distinguish these two mean values indicates the presence of data-dependencies in the leakages. For this purpose, and in order to determine whether some difference observed in practice is meaningful, Welch’s t-test is applied (which is a variant of Student’s t-test that considers different variances and sample

size for the sets $\mathcal{L}_f$ and $\mathcal{L}_r$). The statistic to be tested is defined as:

$$\Delta(\tau) = \frac{\hat{\mu}_f(\tau) - \hat{\mu}_r(\tau)}{\sqrt{\frac{\hat{\sigma}_f^2(\tau)}{N_f} + \frac{\hat{\sigma}_r^2(\tau)}{N_r}}},$$

where $\hat{\sigma}_f^2(\tau)$ (resp. $\hat{\sigma}_r^2(\tau)$) is the estimated variance over the N_f (resp. N_r) samples of $\mathcal{L}_f$ (resp. $\mathcal{L}_r$). Its p-value, i.e. the probability of the null hypothesis which assumes $\Delta(\tau) = 0$, can be computed as follows:

$$p = 2 \times (1 - \text{CDF}_t(|\Delta(\tau)|, \nu)),$$

where CDF_t is the cumulative function of a Student's t distribution, and ν is its number of freedom degrees, which is derived from the previous means and variances as: $\nu = (\hat{\sigma}_f^2/N_f + \hat{\sigma}_r^2/N_r)/[(\hat{\sigma}_f^2/N_f)/(N_f - 1) + (\hat{\sigma}_r^2/N_r)/(N_r - 1)]$. Intuitively, the value of ν is proportional to the number of samples N_f and N_r . When increasing, Student's t distribution gets closer to a normal distribution $\mathcal{N}(0, 1)$.

2.5 Conclusion

In this first chapter, the basic concepts and statistics that are necessary for a complete understanding of the rest of the thesis are introduced. The block cipher and more specifically the AES are introduced as the case-study we use for validating our experiments. Side-channel analysis and standard DPA are then described, as well as the possible counter-measures. Next, we list the statistics and metrics we make use of and describe how they are used in the context of a practical side-channel analysis flow. This flow is referred to for setting the context of our experiments.

Chapter 3

Leakage and points-of-interest detection

Leakage detection tests have recently emerged as a convenient solution to perform preliminary (black box) evaluations of resistance against side-channel analysis. Cryptography Research (CRI)’s non-specific (fixed vs. random) t-test (described in Section 2.4.1) is a popular example of this trend [38]. It works by comparing the leakages of a cryptographic (e.g. block cipher) implementation with fixed plaintexts (and key) to the leakages of the same implementation with random plaintexts (and fixed key), thanks to Welch’s t-test [116]. Besides their conceptual simplicity, the main advantage of such tests, that was carefully discussed in [66], is their low *sampling complexity*. That is, by comparing only two (fixed vs. random) classes of leakages, the detection problem is reduced to a very simple estimation task.

The selection of Points-Of-Interest (POIs) in leakage traces is an important (and not very discussed) problem in the application of Side-Channel Analysis (SCA) attacks. When targeting unprotected implementations, the naive strategy that is commonly used in the literature is to test all the time samples independently. It raises two important challenges. First, how to combine these time samples efficiently, in order to maximize the amount of information extracted from each leakage trace? Second, how to extend this technique in the context of masked implementations where the sensitive data are split into d shares manipulated in different clock cycles (as it is typically the case in software), and only the combination of these shares’ leakage reveals key-dependent information – which makes the complexity of an exhaustive analysis grow combinatorially with d ? Solutions to the first problem typically include dimensionality reduction techniques such as PCA and LDA. These tools

(introduced to SCA in [2, 103] and recently revisited in [11, 17]) essentially project the leakage traces into a lower-dimensional subspace that optimizes some *objective function*. Namely, PCA usually maximizes the variance between the mean leakage traces – i.e. the signal of a first-order DPA, while LDA maximizes the ratio between inter-class and intra-class variances – i.e. its Signal-to-Noise Ratio (SNR), essentially. Their main advantage is to provide a principled and intuitive solution to the problem, since the projection (i.e. eigenvectors) they produce indicate the POIs. Yet, they are somewhat limited when moving to masked implementations for which the information lies in high-order statistical moments, since their objective function is based on a definition of signal that primarily captures first-order leakages¹. Solutions to the second problem are even sparser. To the best of our knowledge, the usual reference for selecting POIs for masked implementations is the educated guess proposed by Oswald et al. in [72] (i.e. an exhaustive search over all d -tuples of time samples in a window that is selected with engineering intuition). Next, Reparaz et al. proposed an alternative solution exploiting Mutual Information Analysis (MIA) [36] that allows gaining a constant (but practically meaningful) factor corresponding to the number of key hypotheses in the attack [91]. In both cases, the proposed tools do not output a projection but a list of the most useful POIs (i.e. d -tuples) in function of the (non-profiled) attack considered.

In this chapter, we want to push the understanding of leakage detection tests one step further, by underlining more precisely their pros and cons, and clarifying their connection with the problem of detecting points-of-interest in leakage traces. As clear from [37], those two problems are indeed related, and one can also exploit t-tests for the detection of POIs in leakage traces. For this purpose, we introduce an alternative test that is based on the CPA distinguisher, the ρ -test. In parallel, we investigate the use of *Projection Pursuits* (PPs), that are optimization algorithms, for the POI detection process. We show how they can easily be instantiated in order to answer the two aforementioned questions. Finally, we show how the ρ -test that is introduced as a leakage detection test, can be used in order to extend the POI detection results for high-order DPA attacks to a non-profiled setting.

¹Of course, a trivial solution would be to apply PCA/LDA to “product traces” containing all the possible products of d -tuples, but this rapidly leads to unrealistic memory requirements in the masked software context that we consider next.

Related publications

1. François Durvaux and François-Xavier Standaert. From leakage detection to the detection of points of interests in leakage traces. Currently submitted, 2015.
2. François Durvaux, François-Xavier Standaert, Nicolas Veyrat-Chavillon, Jean-Baptiste Mairy, and Yves Deville. Efficient selection of time samples for higher-order DPA with projection pursuits. To appear in the proceedings of *COSADE 2015*, 2015.

3.1 Understanding the leakage detection

The main factor influencing the intuitions that one can extract from leakage detection is the implicit assumptions that we make about the partitioning of the leakages (aka leakage model). In the first part of our work, we notice that CRI’s fixed vs. random t-test is one extreme in this direction (since it relies on a partitioning in two classes), which is reminiscent of Kocher’s single-bit DPA (with difference of mean distinguisher, Section 2.3.3). For comparison purposes, we start by specifying an alternative leakage detection test based on the popular Correlation Power Analysis (CPA) distinguisher. The resulting ρ -test directly derives from the hypothesis tests for CPA provided in [64], and relies on a partitioning into 2^n classes where n is the bitsize of the fixed portion of plaintext in the test. We then compare the t-test and ρ -test approaches, both in terms of sampling complexity and based on their *exploitability*.² That is, does a positive answer to leakage detection imply exploitable leakage, and does a negative answer to leakage detection imply no exploitable leakage? Our experimental analysis based on real and simulated data leads to the following observations:

- First, the *sampling complexity* of the t-test is (on average) lower than the one of the ρ -test, as previously hinted [38, 66]. Interestingly, we show that the sampling complexity ratio between the two tests can be simply approximated as a function of a signal-to-noise ratio for the leakage partition used in these tests. This underlines that the difference between the tests is mainly due to their different leakage assumptions, i.e. it is independent of the statistical

²One could also compare the computational complexity of the tests. Since they are based on simple statistics, we will assume that both the t-test and ρ -test can be implemented efficiently. Besides, a minor advantage of the ρ -test is that it can be implemented in a known-plaintexts scenario (vs. chosen-plaintext for the t-test).

test used (backing up the conclusions of [65] for “standard DPA attacks”).

- Second, the *exploitability* of the tests is quite different. On the one hand, leakages that are informative (and therefore can be detected with the ρ -test) but cannot be detected with the t-test are easy to produce (resp. can be observed in practice). Take for example a fixed class of which the mean leakage is the same as (resp. close to) the mean leakage of the random class. On the other hand, the fixed vs. random t-test leads to the detection of many time samples spread around the complete leakage traces. Hence, not all of these samples can be exploited in a standard DPA (because of the diffusion within the cipher).

Concretely, these observations refine the previous analysis in [66], where it was argued that leakage detection is a useful preliminary to white box (worst-case) security evaluations such as advertised in [105]. This is indeed the case. Yet, certain leakage detection tests are more connected with the actual security level of a leaking implementation. In this respect, the fixed vs. random t-test is the most efficient way to perform leakage detection. And the minor drawback regarding its inability to detect certain leakages (e.g. our example with identical means) is easily mitigated in practice, by running the test on large enough traces, or for a couple of keys (as suggested in [38]). By contrast, the main price to pay for this efficiency is a loss of intuition regarding (i) the localisation of the leakage samples that are exploitable by standard DPA, and (ii) the complexity of a side-channel attack taking advantage of the leakage samples for which the detection test is positive. In this respect, the ρ -test can be viewed as a perfect complement, since it provides these intuitions (at the cost of higher sampling complexity).

Next, we show that our reasoning based on the SNR not only allows a better statistical understanding of leakage detection, but can also lead to more efficient t-tests. Namely, it directly suggests that if the evaluator’s goal is to minimize the number of samples needed to detect data-dependent information in side-channel measurements, considering a partitioning based on two fixed plaintexts (rather than one fixed and one random plaintext) leads to significantly faster detection speeds. This is both due to an improved signal (since when integrated over large execution times, samples with large differences between the two fixed classes will inevitably occur) and a reduced noise (since the random class in CRI’s t-test implies a larger algorithmic noise that is cancelled in our proposal). We also confirm these intuitions experimentally, with two representative AES implementations: an 8-bit software

one and a 128-bit hardware one. In both cases, we exhibit detections with roughly 5 times less measurements than when using the previous fixed vs. random non-specific t-test. We believe these results are highly relevant to evaluation laboratories since (i) they lead to reductions of the measurement cost of a leakage detection by a large factor (whereas improvements of a couple of percents are usually considered as significant in the side-channel literature), and (ii) they imply that a device for which no leakages have been detected with one million measurements using a fixed vs. random t-test could in fact have detectable leakages with 200,000 (or even less) measurements.

Specific measurement setup

Our experiments are based on measurements taken as described in Section 2.3.1. Yet, for this part of the work, in each of our experiments, the 128-bit AES master key remains the same for all the measurements as $\kappa = s_0 || s_1 || \dots || s_{15}$, where the s_i 's represent the 16 key bytes. Another difference is that we take measurements of a complete AES execution. When evaluating the fixed vs. random t-test, we build sets of 2000 traces divided into two subsets of 1000 traces each, one corresponding to a fixed plaintext and key, the other corresponding to random plaintexts and a fixed key, denoted as $\mathcal{L}_f$ and $\mathcal{L}_r$ respectively. When evaluating the correlation-based test, we build single sets of 2000 traces $\mathcal{L}$, corresponding to random plaintexts and a fixed key.

In Section 3.1.3, we additionally consider a hardware implementation of the AES of which the design is described in [50]. The same amount of measurement as for the Atmel case are taken, based on a prototype chip embedding an AES core with a 128-bit architecture requiring 11 cycles per encryption, implemented in a 65-nanometer low power technology, running at 60 MHz and sampled at 2 GHz.

3.1.1 A correlation-based leakage detection test

We start by describing an alternative leakage detection test based on the CPA distinguisher, inspired from the hypothesis test described in [64], and taking further advantage of the cross-validation techniques introduced in Section 2.3.4. For k -fold cross-validation, the set of acquired traces $\mathcal{L}$ is first split into k (non overlapping) sets $\mathcal{L}^{(i)}$ of approximately the same size. We then define the profiling sets $\mathcal{L}_p^{(j)} = \bigcup_{i \neq j} \mathcal{L}^{(i)}$ and the test sets $\mathcal{L}_t^{(j)} = \mathcal{L} \setminus \mathcal{L}_p^{(j)}$. Based on these notations, our ρ -test is defined as follows, for a target plaintext byte variable X . First, and for each cross-validation set j with $1 \leq j \leq k$, a model is estimated:

$\hat{\text{model}}_\tau^{(j)}(X) \leftarrow \mathcal{L}_p^{(j)}$. For n -bit plaintext bytes, this model corresponds to the sample means of the leakage sample τ corresponding to each value of the plaintext byte, i.e. $\hat{\mu}_x^{(j)}(\tau)$.³ Next, the correlation between this model and the leakage samples in the test sets $\mathcal{L}_p^{(j)}$ is computed as follows:

$$\hat{r}^{(j)}(\tau) = \hat{\rho}(L_X^{(j)}(\tau), \hat{\text{model}}_\tau^{(j)}(X)).$$

The k cross-validation results $\hat{r}^{(j)}(\tau)$ can then be averaged in order to get a single (unbiased) result $\hat{r}(\tau)$ obtained from the full measurement set $\mathcal{L}$. Following, and as in [64], Fisher's z-transformation is applied to obtain:

$$\hat{r}_z(\tau) = \frac{1}{2} \times \ln \left(\frac{1 + \hat{r}(\tau)}{1 - \hat{r}(\tau)} \right).$$

By normalizing this value with the standard deviation $\frac{1}{\sqrt{N-3}}$, where N is the size of the evaluation set $\mathcal{L}$, we obtain a sample that can be (approximately) interpreted according to a normal distribution $\mathcal{N}(0, 1)$. This allows us to compute the following p-value for a null hypothesis assuming no correlation:

$$p = 2 \times (1 - \text{CDF}_{\mathcal{N}(0,1)}(|\hat{r}_z(\tau)|)),$$

where $\text{CDF}_{\mathcal{N}(0,1)}$ is the cumulative function of a standard normal distribution. Besides exploiting cross-validation (which allows us to obtain unbiased estimates for Pearson's correlation coefficient), the main difference between this test and the hypothesis test in [64] is that our model is built based on a plaintext byte rather than a key-dependent intermediate value. This allows us to implement it in a black box manner and without key knowledge, just as the previous t-test.

3.1.2 Comparison with the *fixed vs. random* test

In order to discuss the pros and cons of the two leakage detection tests, we now consider various experimental results. We start with a simulated setting which allows us to control all the parameters of the leakages to detect, in order to discuss the sampling complexity of both methods. Next, we analyze actual leakage traces obtained from a measurement setup similar to the one described in Section 2.3.1, which allows us to put forward the intuitions provided by the t-test and ρ -test regarding the time localization of the informative samples in our traces.

³If there is no available trace for a given value of x , which happens when the evaluation set is small, the model takes the mean leakage taken over all the traces in $\mathcal{L}_p^{(j)}$.

Simulated experiments

We define a standard simulated setting for the leakages of a block cipher, where an intermediate computation $z = S(y = x \oplus s)$ is performed, with S an 8-bit S-box. It gives rise to a (multivariate) leakage variable of the form:

$$L_X = [\text{HW}(X) + R_1, \text{HW}(Y) + R_2, \text{HW}(Z) + R_3],$$

where HW is the Hamming weight function, R_1, R_2 and R_3 are Gaussian distributed random noises with mean 0 and variance σ_n^2 , and the index X recalls that in our detection setup, the evaluator only varies the plaintext. For t-tests, the set $\mathcal{L}_f$ contains leakages corresponding to fixed values of x, y or z , denoted as x_f, y_f, z_f , while the set $\mathcal{L}_r$ corresponds uniformly random x 's, y 's or z 's. For ρ -tests, the leakages all correspond to uniformly random x 's, y 's or z 's.

Concretely, we analyzed the t-test based on the third sample of L_X (which corresponds to the target intermediate value z), and for different fixed values of this intermediate value. This choice is naturally motivated by the counter-example given in introduction. That is, since the average leakage of the random class equals 4 in our simulation setting, a fixed class such that $\text{HW}(z_f) = 4$ should not lead to any detection. And extending this example, the bigger the difference between $\text{HW}(z_f)$ and 4, the easier the detection should be.

In parallel, we investigated the ρ -test in two different scenarios. First the realistic case, where the model estimation using k -fold cross-validation described in Section 3.1.1 is applied (using the standard $k = 10$). Second, a theoretical simplification where we assume that the evaluator knows the perfect (here Hamming weight) model, which implies that all the samples in the set $\mathcal{L}$ are directly used to compute a single estimate for the correlation $\hat{r}(\tau) = \hat{\rho}(L_X(\tau), \text{model}_\tau(X))$.

The results of our experiments are given in Figure 3.1, where the upper part corresponds to a noise variance $\sigma_n^2 = 50$ and the lower part to a noise variance $\sigma_n^2 = 100$. In both cases, we set the detection threshold to 5, which is the value suggested in [6]. They allow the following relevant observations:

1. *On the impact of the noise.* As doubling the noise variance generally doubles the measurement complexity of a side-channel attack, it has the same impact on the sample complexity of a leakage detection test. For example, detecting a difference between a fixed class such that $\text{HW}(z_f) = 2$ and a random class with the t-test requires ≈ 1300 traces in the upper part of the figure and ≈ 2600 traces in its lower part. Similar observations hold for all the tests.

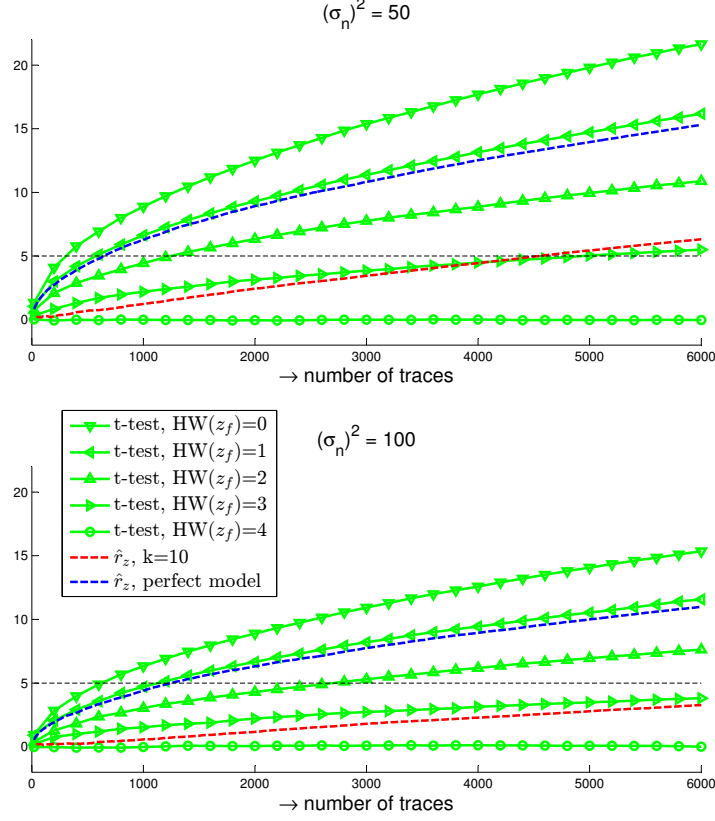


Figure 3.1: Leakage detection on simulated traces, Hamming weight leakage function.

2. *On the impact of the fixed value for the t-test.* As expected, for both σ_n^2 , a fixed class such that $\text{HW}(z_f) = 4$ cannot be distinguished at all from the random class (since they have the same mean). By contrast, a fixed class such that $\text{HW}(z_f) = 0$ is extremely fast to distinguish from the random class.
3. *The ρ -test can have larger sampling complexity.* This naturally depends on the fixed value for the t-test. But assuming that several samples from a trace are used in a the leakage detection (which is usually the case, as will be shown in our following measured experiments), some of them should lead to faster leakage detection with the t-test than with the ρ -test.

4. *It's all in the SNR.* That is, just as in standard DPA, the sampling complexity of a detection test essentially depends on the SNR of its leakage partitioning. For the ρ -test, we can directly exploit Mangard's definition from CT-RSA 2004 for this purpose [62]. That is, the signal corresponds to the variance of the random variable $\text{HW}(Z)$ with Z uniform, which equals 2 for 8-bit values, and the noise variance equals to σ_n^2 . As for the t-test, we need to define an binary random variable B that is worth $\text{HW}(z_f)$ with probability 1/2 and $\text{HW} = 4$ with probability 1/2. For each value of the fixed z_f , the signal then corresponds to the variance of B , and the noise variance equals to σ_n^2 for the fixed class, and $\sigma_n^2 + 2$ for the random class (since in this case, the noise comes both from the variable Z and from the noise R). For example, this means a signal 0 for the fixed class $\text{HW}(z_f) = 4$, a signal 0.25 for the fixed class $\text{HW}(z_f) = 3$, a signal 1 for the fixed class $\text{HW}(z_f) = 2$, a signal 2.25 for the fixed class $\text{HW}(z_f) = 1$, and a signal 4 for the fixed class $\text{HW}(z_f) = 0$. Ignoring the small noise differences between the tests, it means that the sampling complexity for detecting leakages with the t-test and a fixed class $\text{HW}(z_f) = 1$ should be close to (and slightly smaller than) the sampling complexity for detecting leakages with the ρ -test. And this is exactly what we observe on the figure, for the ρ -test with a perfect model (see next). Note that the same reasoning can be used to explain the sampling complexities of the t-test for different fixed values. For example, the case $\text{HW}(z_f) = 3$ requires four times as many traces than the case $\text{HW}(z_f) = 2$ on the figure.

An important consequence of this observation is that, as for standard DPA attacks, the choice of statistic (here the t-test or ρ -test) has limited impact on the sampling complexity. For example, one could totally design a ρ -test based on a partition in two (fixed and random) classes, that would then lead to very similar results as the t-test (up to statistical artifacts, as discussed in [65]).

5. *Estimating a model can only make it worse.* Besides the potentially lower signal, another drawback of the 256-class ρ -test from the sampling complexity point-of-view is that it requires the estimation of a model made of 256 mean values. This further increases its overheads compared to the t-test, as illustrated in Figure 3.1 (see the $\hat{r}_z$ curve with $k = 10$ -fold cross-validation). In this respect, we first note that considering larger k 's only leads to very marginal improvements of the detection (at the cost of significant computational overheads). Besides, we insist that this estimation

is unavoidable. For example, ignoring the cross-validation and testing a model with the same set as its profiling set would lead to overfitting and poor detection performances. In other words, it is the size of the partition used in the ρ -test that fixes its SNR (as previously discussed) and estimation cost, and both determine the final sampling complexity of the test.

Note that the above conclusions are independent of the leakage function considered (we repeated the same experiments with identity rather than Hamming weight leakages, and reached the same conclusions). Therefore, these simulated results confirm our introduction claim that for leakage detection only, a fixed vs. random t-test is the preferred method, and that their gains over a ρ -test can be easily predicted from a leakage function/partition and its resulting SNR metric.

Measured experiments

We now extend the previous simulated analysis to a practically-relevant case of actual AES measurements, obtained from the setup described in Section 2.3.1. We will divide our investigations into two parts. First, a global analysis will consider the leakage traces of the full AES executions, in order to discuss the sampling complexity and intuitions regarding the POIs for our two detection tests. Next, a local analysis will be used in order to discuss possible false negatives in the t-test, and intuitions regarding the informativeness of the detected samples.

Global analysis. The results of a fixed vs. random t-test and a ρ -test for leakage traces corresponding to an entire AES Furious execution are provided in Figure 3.2, from which two main observations can be extracted:

1. *The t-test has lower sampling complexity on average.* This is essentially the concrete counterpart of observation (3) in the previous section. That is, we already know that for some fixed values of the plaintext, the t-test should have a lower sampling complexity. Figure 3.2 confirms that when looking at complete AES traces, those “easy-to-detect” fixed values are indeed observed (which is natural since the AES Furious implementation accounts for a bit more than 3000 clock cycles, and the intermediate values within such a block cipher execution should be uniformly distributed after a couple of rounds). Concretely, this means that the sampling complexity for detecting leakages with a similar confidence increases from ≈ 200 traces for the t-test to ≈ 2000 traces for the ρ -test, i.e. a factor

≈ 10 which is consistent with the previous simulations. Note that even in the context of a hardware implementation with a reduced cycle count (e.g. 11 cycles per AES execution), finding fixed values that are easy-to-detect for the t-test is feasible by trying a couple of fixed plaintexts and keys.

2. *The ρ -test (resp. t-test) does (resp. not) provide intuitions regarding exploitable leakage samples.* This is easily seen from the figure as well. Whereas the t-test detects information leakage everywhere in the trace, the ρ -test is much more localized, and points towards the samples that depend on the single plaintext byte that is varying. Since the key is fixed in leakage detection, it implies that peaks are observed whenever this (useless) plaintext byte and the (useful) intermediate values that bijectively depend on it are manipulated, e.g. the key addition and S-box outputs in Figure 3.2. In other words, the ρ -test is mostly relevant for the detection of POIs that are exploitable in a standard DPA attack (i.e. excluding the false positives corresponding to plaintext manipulations).

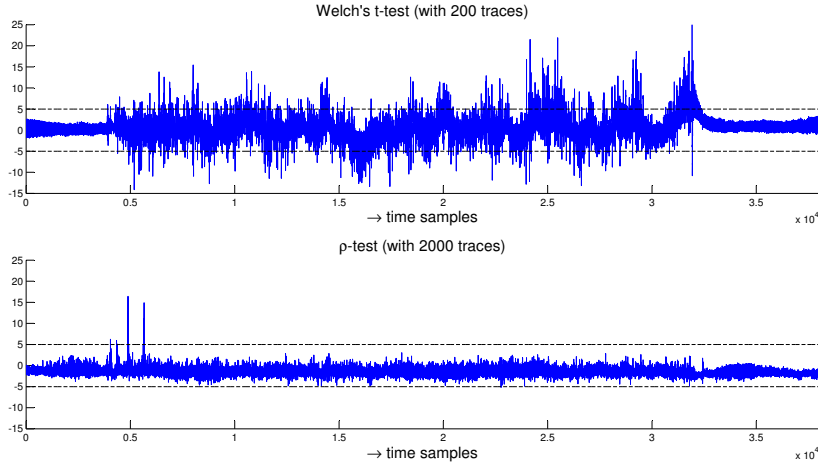


Figure 3.2: Leakage detection on real traces, entire AES execution.

Local analysis. The results of a fixed vs. random t-test and a ρ -test for leakage traces corresponding to the beginning of the first AES round

execution are provided in Figure 3.3, from which two main observations can be extracted.⁴

1. *Hard-to-detect leakage samples for the t-test can be observed.* More precisely, the lower part of Figure 3.3 exhibits three peaks which exactly correspond to the manipulation of a plaintext byte (first peak), the key addition (second peak) and the S-box execution (third peak), just as the three samples of our simulated setting. Knowing that our Atmel implementation of the AES has leakages that can be efficiently exploited with a Hamming weight model (as in our simulations) [104], we selected the fixed plaintext byte of the t-test such that $\text{HW}(z_f) = 4$. As illustrated in the upper part of the figure, the leakages of this fixed intermediate value are indeed difficult to tell apart from the ones of its random counterpart. More precisely, the ρ -test clearly exhibits a peak for this intermediate value after 2000 traces, which does not exist in the t-test experiment using a similar sampling complexity. Whereas we cannot exclude that such a peak would appear for a larger number of traces (since the chip does not exactly follow the Hamming weight leakage model), this confirms that not all leakage samples are easier to detect with the t-test than with the ρ -test.
2. *The ρ -test does provide intuitions regarding the informativeness of the leakage samples.* Eventually, a straightforward advantage of the ρ -test is that the value of its correlation coefficient estimates brings some intuition regarding the complexity of a side-channel attack exploiting this sample, which is only provided up to a limited extent by the t-test. Indeed, a side-channel attack exploiting an n -bit intermediate value is most efficient if it relies on an n -bit model, as considered by the ρ -test (otherwise $n - 1$ bits out of n will produce “algorithmic noise”). In this context, we can take advantage of the connection between Pearson’s correlation coefficient and the information theoretic metrics in [105] (see [65]), themselves related to the worst-case complexity of standard DPA attacks [29].

3.1.3 Improved leakage detection test

One central conclusion of the previous section is that the sampling complexity of leakage detection tests highly depends on the SNR of the

⁴Exceptionally for this experiment, we considered a single varying byte for the t-test, in order to better exhibit intuitions regarding the detected samples for a single S-box.

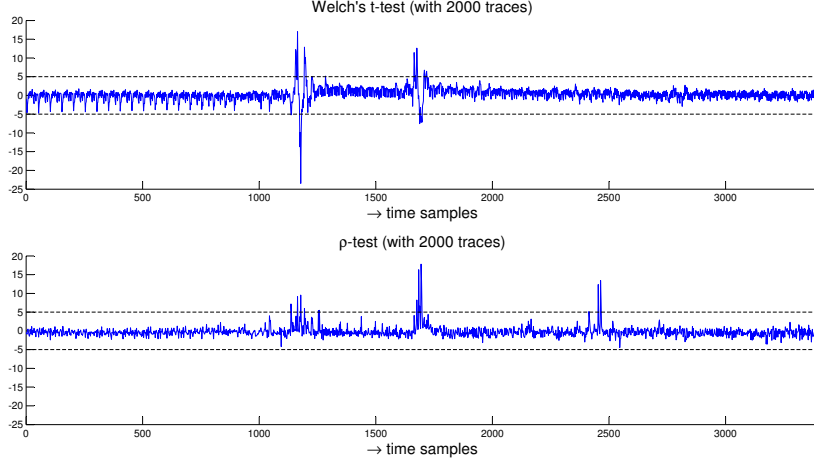


Figure 3.3: Leakage detection on real traces, first-round AES key addition and S-box.

leakage partition on which they are based. Interestingly, this observation directly suggests a natural improvement of CRI's non-specific (fixed vs. random) t-test. Namely, rather than performing the test based on a fixed and a random class, a more efficient solution is to perform a similar test based on two fixed classes (i.e. two fixed plaintexts). On the one hand, this directly reduces the detection noise from $2\sigma_n^2 + \sigma_{\text{alg}}^2$ to $2\sigma_n^2$, since it cancels the algorithmic noise due to the variations of the random class. Taking the example of Hamming weight leakages, this algorithmic noise corresponds to $\sigma_{\text{alg}}^2 = 2$ for 8-bit values, but it increases for larger parallel implementations (e.g. it is worth $\sigma_{\text{alg}}^2 = 32$ for 128-bit implementations). On the other hand, and when applied to large traces, such a partitioning also increases the signal with high probability, for the same argument as used to avoid false positives in CRI's t-test (i.e. by applying the detection to large enough traces, large differences between the two fixed classes will inevitably occur). Taking the example of Hamming weight leakages again, we can easily compute the probability (over random inputs) that a certain leakage difference is obtained for both types of partitions (i.e. fixed vs. random and fixed vs. fixed), and the resulting signal variance, as illustrated in Figure 3.4. We conclude from this figure that (i) the fixed vs. fixed partitioning allows reaching larger differences (so larger signals) and (ii) the fixed vs. fixed

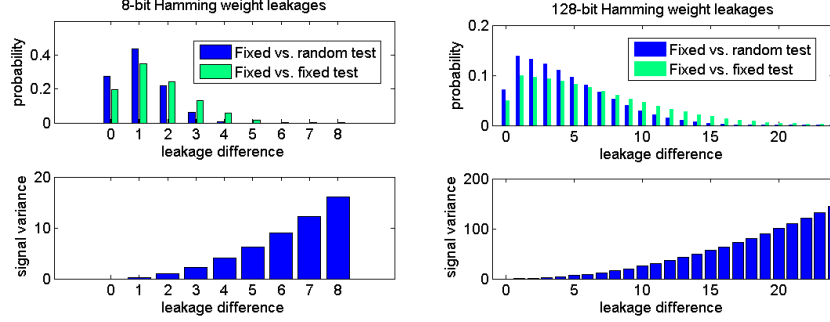


Figure 3.4: Fixed vs. random and fixed vs. fixed leakage detection signal.

partitioning allows doubling the average signal (i.e. the dot product of the probabilities and variances in the figure). So both from the noise variance and the (best-case and average case) signal points-of-views, it should improve the sampling complexity of the detection test.⁵ In other words, a leakage detection based on a fixed vs. fixed leakage partition should theoretically have better sampling complexity than with a fixed vs. random one.

Quite naturally, the exact gains of this new detection test depend on the actual leakages. So as in the previous section, we confirmed our expectations with two case studies. First, we compared the fixed vs. random and fixed vs. fixed t-tests based on our software AES implementation. The results of this experiment are in Figure 3.5 where we observe that data-dependent leakages are detected with similar confidence with approximately 5 times less traces thanks to our new partitioning. Next, we investigated the context of the hardware implementation of the AES described in introduction of this section. As illustrated in Figure 3.6, similar gains are obtained. Note however that despite we gain an approximate factor 5 in both cases, the reasons of this gain are different. Indeed, the software implementation case is dominated by an increase of signal (due to its large cycle count) and has limited algorithmic noise. By contrast, the hardware implementation has larger algorithmic noise (corresponding to 128-bit random values) but less improvements of the signal (because its traces are only 11-cycle long). Even larger gains could be obtained by combining both the signal and noise effects (e.g. by considering multiple keys for the hardware implementation). Based

⁵A similar conclusion can be obtained for other leakage functions, though the binomial distribution of the Hamming weight leakages naturally make computations easier.

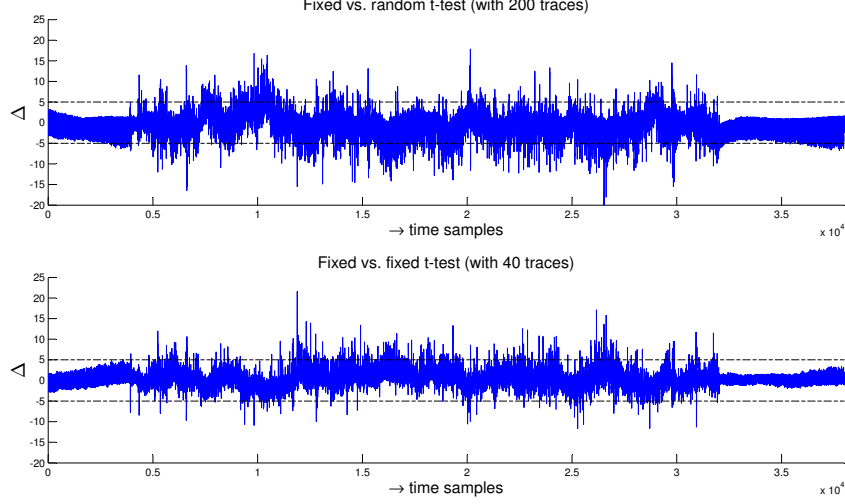


Figure 3.5: Improved leakage detection on real traces (Atmel implementation).

on these theoretical arguments and experimental confirmation, we expect our fixed vs. fixed partitioning to lead to faster leakage detections in most practical scenarios.

3.1.4 Summary & open problems

The discussion in this section highlights that there are significant differences between current approaches to side-channel security evaluation. On the one hand, CRI’s Test Vector Assessment Methodology (TVLA) aims at minimizing the evaluator’s efforts. Very concretely, non-specific t-tests as proposed in [21, 38] are indeed good to detect univariate and first-order leakages. As we observed in Section 3.1.3, slightly tweaking the selection of the classes (from fixed vs. random to fixed vs. fixed) allows significantly improving the detection speed in this case. In this respect, we note that our investigations focused on plaintext variations (allowing non-profiled detections), but similar conclusions hold for other types of dependencies (e.g. one could consider two fixed keys with the same plaintext, or two different pairs of plaintext and key). Despite minor theoretical caveats (i.e. the possibility of false positives and negatives), the application of such 2-class t-tests turns out to be extremely efficient. On the other side of the spectrum, complete (ideally worst-case)

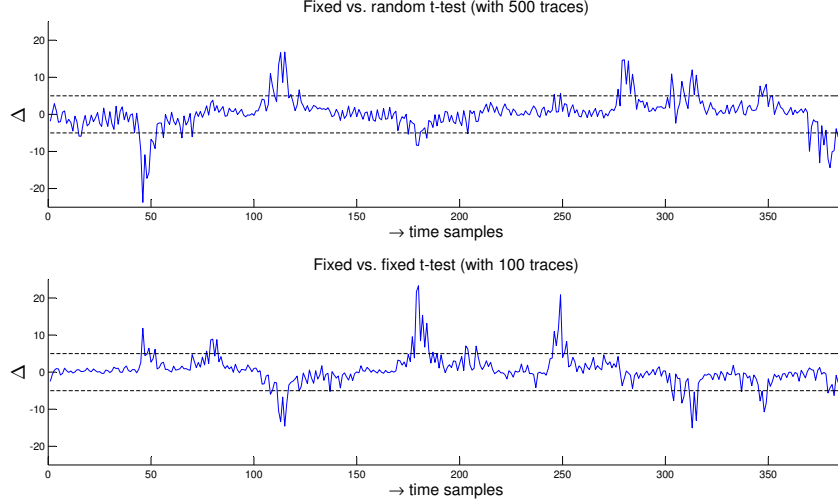


Figure 3.6: Improved leakage detection on real traces (ASIC implementation).

security evaluations such as discussed in [105] rather aim at a precise rating of the security level, possibly considering the adversary’s computing power [112], which is an arguably more expensive task. In this case, the selection of POIs is usually a necessary first step. As also discussed in this section, and when restricted to univariate and first-order leakages, the main reason for the additional cost of this approach (including the selection of POIs) is the larger number of classes for which the leakage distribution has to be well estimated. In this context as well, our investigations focused on non-profiled POI detection (which can be performed efficiently for the first/last cipher rounds). But similar conclusions hold in the profiled evaluation setting, which allows finding POIs in all the cipher rounds, and is necessary for worst-case analysis.

These different methodologies naturally raise the question of which one to use in which context, and whether they can be connected to some extent, leading to the following open problems. First, how to generalize (simple) detection tests to capture more types of leakages? Moving from univariate first-order leakages to univariate higher-order leakages appears reachable with existing tools. One option is to exploit more general statistical tests, e.g. the mutual information based one in [66]. Another option is to work “by moments” and to test higher-order moments of the leakage distributions (with t-test, F-tests, ..., if 2 classes

are considered, with Moments-Correlating DPA or equivalent tools if more classes are considered). Moving to multivariate leakage detection appears much more difficult. At least, testing all pairs/triples/... of samples in a trace rapidly turns out to be infeasible as the size of the traces increase, which usually leads current evaluations to be based on heuristics (e.g. further discussed in Section 3.2). Second, can we extrapolate or bound the worst-case security level of an implementation based on simple statistical tests? For example, the recent work in [29] shows that one can (in certain well-defined conditions) bound the security level of an implementation, measured with a success rate and in function of the number of measurements and computing power of the adversary, based on information theoretic metrics (such as the mutual information in general, and the SNR if we only consider univariate attacks). But as discussed in this section, evaluating an SNR is still significantly more expensive than detecting leakages with non-specific tests. So of course, it would be interesting to investigate whether it is possible to bound the security level based on simpler leakage detection tests. In case of negative answer, it anyway remains that such leakage detection tests can always be used as a preliminary to more expensive approaches (detecting POIs, security evaluations), e.g. to reduce the dimensionality of the traces.

3.2 Selecting time samples with projection pursuits

In this section, we investigate the use of *Projection Pursuits* (PPs), as alternative tools for the selection of POIs in leakage traces [34]. Intuitively, PPs machine-pick “interesting” low-dimensional projections of a high-dimensional data space by numerically maximizing a certain objective function. They essentially work by tracking the improvements (or lack thereof) of the projection when applying small random modifications. Their main advantage in our context is that they can deal with any objective function, which naturally fits the problem of higher-order SCA. Their main drawback is (in general) their heuristic nature, since the convergence of the method is not guaranteed and its complexity is context-dependent. As a result, and in order to validate the interest of PPs in our SCA context, we first applied them to the simple case of an unprotected implementation of the AES. We show that different objective functions can be efficiently used for this purpose, leading to powerful subspace-based attacks, with similar informativeness as previous solutions such as LDA.

Next, we moved to the more challenging context of masking. In this case, we combined the (linear) projection with an objective function exploiting higher-order statistical moments. Initial experiments suggest that the straightforward implementation of a PP algorithm is not efficient at detecting the POIs of such protected implementations (especially as the number of useless dimensions in the traces increases). The main reason is that as long as a d -tuple of POIs is not present in the projection, the objective function essentially returns random indications. Interestingly, we then show that a specialized PP algorithm exploiting an improved *local search* could give excellent results even in this challenging context. Intuitively, it works by looking for the best size and position of d windows covering parts of the traces, again by iterating small random perturbations. Our experiments suggest that we can recover POIs with significantly less calls to the objective function than an exhaustive analysis. We further discuss the main parameters influencing the success of such a detection method, and detail the time vs. measurement complexity tradeoff resulting from these parameters.

Cautionary note & related works. In general, a projection search algorithm can be evaluated according to two orthogonal axes, namely its time and data complexity (i.e. how many iterations and measurements do we need to obtain a projection?) or the informativeness of its outputs (which relates to the data complexity of an attack exploiting the projections obtained). Hence, it is worth recalling first that (standard) dimensionality reductions for unprotected implementations indeed optimize informativeness, whereas existing solutions to detect POIs in masked implementations focus on the complexity issue (because of the more challenging nature of the problem). In this context, we note that comparing different projection informativeness (e.g. PCA, LDA and the recent works in [43, 71]), in the context of unprotected implementations, is of limited interest anyway. Indeed, it has been shown in [65] that the objective functions of LDA (which improves over PCA in terms of informativeness) and [43, 71] are essentially equivalent in this case, meaning that LDA, these works and our new projections all have similar informativeness as well (up to statistical artifacts). Nevertheless, and for completeness, we show empirical evidence of the gain they provide over PCA and its impact in the DPA contest v2 [75]. As for comparisons in the case of higher-order leakages and masked implementations, the main issue is that none of the previous dimensionality reductions generalizes to such contexts⁶. In fact, and as witnessed by the previous state-of-

⁶More precisely, the results in [43, 71] are actually similar to ours in the first-order setting. In fact, they can be viewed as a heuristic (computationally efficient) analogue

the-art, existing solutions for detecting POIs in masked implementations are mainly based on exhaustive approaches, which turns out to be too intensive as the size of the leakage traces increases. Hence, our main contribution is to provide a first efficient alternative for dealing with large traces containing only higher-order information leakages. For this purpose, and since our focus is on complexity issues, we will put ourselves in the most challenging scenario, i.e. a black box analysis where no information about the source code is available, and compare the gains of our optimizations over combinatorial search and other naive approaches such as signal integration [18]. Quite naturally, any engineering intuition allowing an educated guess – by focusing only on certain parts of the traces – could be exploited as well. Besides we will also discuss the differences between our work and the one of Reparaz et al. and exhibit that they are essentially complementary.

3.3 PPs against unprotected devices

In this section we investigate the application of PPs to the simple case of the (unprotected) AES Furious implementation (Sections 2.1.4 and 2.3.1). In this context, our goal is to find a projection vector α that will convert the N_s samples of a leakage vector $\mathbf{l}_y^i$ to a single (projected) sample λ_y^i , that is:

$$\lambda_y^i = \sum_{\tau=0}^{N_s-1} \alpha(\tau) \cdot l_y^i(\tau)$$

such that univariate attacks exploiting the λ_y^i 's will be most efficient. This essentially requires defining an objective function that measures the “informativeness” of these samples. As previously mentioned, this task is quite easy when first-order information is available in the leakage traces: Pearson’s correlation coefficient obtained from a CPA and Mangard’s SNR (Section 2.3.3) are natural candidates – we will try them both in our experiments. Following the equivalence results in [65], they should provide similar results in this case (also similar to the ones that would be obtained with an information theoretic metric).

to LDA. But their application to the higher-order case would be difficult for the same reasons as mentioned at the beginning of Section 3.4 for our (non-specialized) projection search. In this respect, an important difference between this previous work and ours is the separation between the objective functions and optimization algorithms: we need to change both in order to efficiently deal with higher-order leakages.

3.3.1 Projection pursuit algorithm

The pseudo-code of our projection pursuit algorithm is given in Algorithm 2. It essentially repeats (N_r times) the selection of a random index r followed by a maximization of the objective function for the corresponding time sample, based on the set of profiling traces $\mathcal{L}^p$ (which contains traces for all the intermediate values y). For this purpose, the *max_search()* function consists in successive parabolic interpolations (illustrated in Figure 3.7), which work in two iterated steps. We first look for samples that enclose the extremum as follows. From a starting point x_1 , we add a Δ in the direction that increases f_{obj} (blue plain curve) to get x_2 . Then, we keep adding Δ 's until finding x_3 such that $y_3 < y_2$ (see Figure 3.7.a). As the weights assigned to each time sample are between 0 and 1, we typically take Δ 's corresponding to a couple of percents (e.g. 0.1 in our experiments) and repeat such additions at most $1/\Delta$ times. Then, based on these three points, we start interpolating (as in the dashed red curve of Figure 3.7.b-c). This process is iterated N_{it} times, during which we replace the “oldest” x -point by the x -coordinate (x_v) of the parabola vertex (y -values are re-computed accordingly). The new $\alpha(t)$ gets its value from the median x -value at the end of the last iteration. In our experiments, $N_{it} = 3$ iterations were enough to get a good approximation of the maximum. This method has the advantage of being very fast to compute and converge. Note finally that the number of repetitions N_r should ideally be larger than the number of samples N_s (e.g. twice, typically), because some weights benefit from being re-adjusted after the modification of other $\alpha(t)$'s. Yet, when applied in the context of an unprotected implementation, the time complexity of Algorithm 2 was never a practical limitation (it typically corresponded to a couple of minutes of computations in our experiments).

Algorithm 2 Projection Pursuit.

```

basic_PP( $N_r, N_{it}$ )
 $\alpha = initialize()$ ;
repeat  $N_r$  times
     $r = rand\_index(N_s)$ ;
     $\alpha_{new} = max\_search(@f_{obj}, \mathcal{L}^p, \alpha, r, N_{it})$ ;
     $\alpha = \alpha_{new}$ ;
end

```

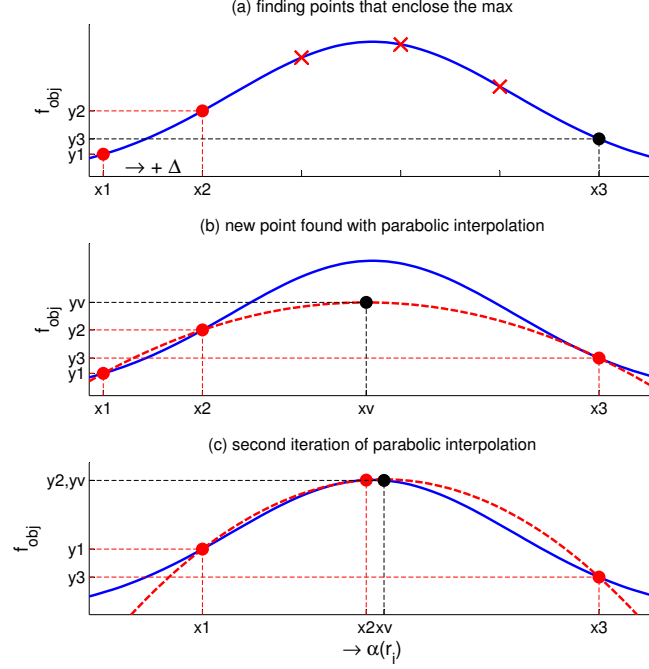


Figure 3.7: Successive parabolic interpolations.

3.3.2 Experimental results

We implemented the PP algorithm for both the CPA and SNR objective functions, and targeted the first AES key byte for illustration. For each of the 256 values of $y = x \oplus s$, we measured $N_p = N_t = 50$ traces for the CPA objective function, and $N_t = 100$ traces for the SNR one, each of them made of $N_s = 1500$ time samples. We set N_r , N_{it} and Δ as just explained (to 3000, 3 and 0.1, respectively). The projections obtained in both cases are given in Figure 3.8, for illustration. As expected, they are very similar. We then computed success rates to compare the quality of the projections obtained with the most informative sample, by performing 2000 experimental univariate Template Attacks (TA). These results show the effectiveness of the projections as they only need 7 traces to get a 90% success rate, against 28 traces for the univariate TA. It also confirms that both objective functions are indeed equivalent in this case. It is finally interesting to compare our findings with the results in [104] that target a similar implementation (with very similar success

rate for the univariate TA). In particular, we see that the univariate attack based on the single sample provided by our projections leads to approximately the same data complexities as the hexavariate template attack taking (heuristic) advantage of all the POIs in this previous work. This informally confirms the quality of our projection.

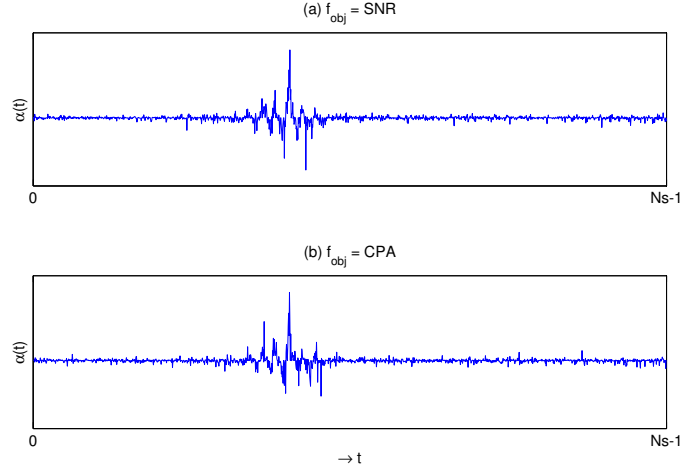


Figure 3.8: Projection profiles

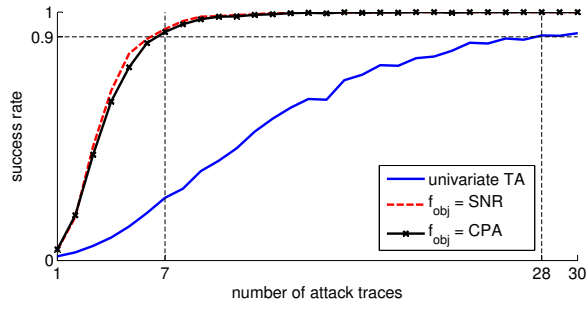


Figure 3.9: Template attack success rates against unprotected device

3.3.3 Discussion & comparison with existing works

As mentioned in introduction, the open literature essentially provides satisfying solutions for detecting POIs and reducing the dimensionality of leakage traces obtained from unprotected devices, and this holds both from the complexity and informativeness points-of-view. Namely, from the complexity point-of-view, testing all time samples exhaustively is usually achievable in this case. And from the informativeness point-of-view, any projection using one of the criteria in [65], e.g. like LDA does with the SNR, should be optimal. In practice though, computing LDA for large traces may lead to numerical issues, as it requires estimating a covariance matrix of the size of the leakage traces (whereas PCA enables a “small sample size variant” where the size of this covariance matrix can be limited to the number of key hypotheses [2]). In this respect, both our projection pursuits in Algorithm 2 and the proposal in [71] provide useful alternatives. Note that the approach in this previous work slightly differs from ours since its experiments are specialized to CPA with a Hamming distance leakage model (hence deviate from our optimality goal). By contrast, we rather suggest to use a profiled correlation coefficient (or the SNR) as objective function(s), since the evaluation of the objective function requires some key knowledge anyway. These minor differences highlight that projection pursuits are versatile and can be applied with a variety of objective functions. Eventually, and despite it is not the main concern in this work, we want to emphasize that deviating from an optimal objective function can lead to substantial differences in the attack success rates. For example, both PCA and our projection pursuit algorithm using the SNR as objective function were tested in the context of the DPA contest v2 [75]. As illustrated in Figure 3.10, the data complexity of an attack using PCA is approximately doubled in this case, which can be explained by the correlation between the noise distributions taken at different time samples. Note that this projection pursuit was included amongst the best attacks of this contest, leading to a data complexity of 1173 traces (439 after key enumeration) as detailed in [19].

3.4 PPs against masked implementations

In contrast with the previous section, detecting POIs in leakage traces of masked implementations (Section 2.2.3) is a quite challenging task. From the complexity point-of-view, exhaustive approaches may grow exponentially with the number of shares (if these shares are manipulated at different time samples), making them unpractical for long traces. Furthermore, the information in the leakages of masked implementation

lies in higher-order moments of their probability distribution, which are harder to estimate. As a result, the direct application of Algorithm 2 with the previous objective functions in this context does not provide successful results. In the (simple) case where the shares of a masking scheme are manipulated in parallel, adapting the objective function may be sufficient to deal with this problem. But in case of software implementations, where the shares are manipulated at different time samples, the algorithm itself has to be adapted. Intuitively, this is because it works by modifying time samples one at a time, while for such masked implementations, we require at least one meaningful d -tuple of samples to be active in the projection for an objective function to output relevant information. We now describe how to specialize PPs to take this constraint into account, and detect POIs for masked implementations.

3.4.1 Specialized projection pursuit algorithm

The main tool used in our following optimization is local search, which is a collection of iterative methods that are efficient for quickly finding good solutions to optimization problems (note that the previous PP algorithm can be viewed as a simple local search). Despite heuristic, it generally works more efficiently than exhaustive analyses. Furthermore, local search has very limited storage requirements. For example, in our context, it exploits the leakage traces directly – which is a significant advantage compared to heuristics exploiting “product traces” as mentioned in the introduction of this section. A good reference to these methods is [47]. Their working principle is simple: they always keep a solution (called the current solution) as well as the best solution found since the beginning of the search. At each iteration of the algorithm, the current solution is perturbed, giving a set of new solutions, called its neighbourhood. One of the neighbouring solutions is then selected and replaces the current solution. The algorithm terminates when its convergence criterion is met (e.g. number of iterations without improvement, time limit, etc.). Intuitively, such an approach to optimization exploits *diversification* and *intensification*. The former aims at exploring a large and diverse search space, while the latter intends to improve the current solution. Their combination is expected to avoid being trapped in local optima.

When applied to masking, one key element has to be taken into account by optimizations. Namely, the sensitive variables are split into d shares and the objective function should not be informative as long as a meaningful d -tuple of shares is not present in the projection. Besides, in practice it frequently happens that dimensions near a POI also contain

valuable information. These two facts motivate the way we designed our improved search algorithm as follows. First, we consider a projection vector containing d windows of non-zero weights (all the others being zero) and denote a group of successive dimensions as a window. The weights inside these windows are uniform. In this context, and since local search only considers local modifications of the current solution, the information given by the objective function will return essentially random indications (so no reliable information) if this current solution does not cover the d shares. On the contrary, when the windows span a d -tuple of shares, the objective function can be used to refine the current solution. For this reason, our specialized PP algorithm will be split into two parts next denoted as `find_sol` and `improve_sol`. The `find_sol` phase probes the search space with large windows and a lot of randomness until it has good indication that the windows span the d -tuples of shares. In order to detect that the windows span these d -tuples, we use two sets of profiling traces ($\mathcal{L}_{tr}^p$ and $\mathcal{L}_{va}^p$, where `tr` stands for training and `va` for validation). Then, the `improve_sol` phase refines those windows. The `find_sol` phase thus puts more emphasis on diversification and the `improve_sol`, on intensification.

Algorithm 3 Specialized projection pursuit algorithm using local search.

```

specialized_PP_Local_Search( $d, W_{len}, T_{det}, TP := TP' \cup TP''$ )
   $\alpha = \text{find\_sol\_phase}(d, W_{len}, T_{det}, TP')$ ;
  if( $\alpha \neq null$ )
    return  $\text{improve\_sol\_phase}(\alpha, TP'')$ ;
  end

```

The pseudocodes of the specialized PP algorithm using local search are given in Algorithms 3, 4 and 5. These algorithms depend on various parameters: some of them will be explicitly discussed as they hold important intuitions, the remaining ones – next denoted as technical parameters (TP) – will be fixed according to state-of-the-art strategies. Our main tool is the `specialized_PP_Local_Search` function (Algorithm 3). As just explained, it organizes the search in two main steps. The first one is the `find_sol` phase which returns a first candidate projection α (after N_r^f repetitions). If this first step is successful, the `improve_sol` phase is repeated N_r^i times to refine the solution. The `find_sol` phase is described in Algorithm 4. At each iteration, it randomly selects d windows of length W_{len} with non-zero weights (function `random_window`).

Algorithm 4 Find solution phase.

```

find_sol_phase( $d, W_{len}, T_{det}, TP'$ )
 $TP' := \{N_r^f, num\_hops\}$ 
 $i = 0$ ;
repeat  $N_r^f$  times
     $\alpha = random\_window(d, W_{len})$ ;
     $neighbourhood = get\_neighbours\_FS(\alpha, num\_hops)$ ;
     $best\_neighbour = max(@f_{obj}, neighbourhood, \mathcal{L}_{tr}^p)$ ;
    if  $f_{obj}(best\_neighbour, \mathcal{L}_{tr}^p) > T_{det}$ 
        if  $f_{obj}(best\_neighbour, \mathcal{L}_{va}^p) > T_{det}$ 
            return ( $i + 1, best\_neighbour$ );
        end
    end
     $i++$ ;
end
end

```

All the neighbours of the solution are then computed with the function `get_neighbours_FS`. Each neighbour is constructed by moving one of the windows left or right (if we see the projection vector as a row vector). The lengths of the moves considered are small multiples of the window length (as set by the `num_hops` parameter). During the computation of the neighbours, the collisions between windows are avoided in order to keep d distinct windows. Next, the best neighbour is selected as the neighbour having the maximal evaluation of f_{obj} on the set $\mathcal{L}_{tr}^p$. This best neighbour is finally tested to detect if a d -tuple of shares is spanned by the windows. The detection is based on a threshold T_{det} on the objective function that will be carefully discussed in the next section. In order to dodge the randomness of the objective function when the d shares are not spanned, this threshold has to be exceeded on both the training and validation sets of traces $\mathcal{L}_{tr}^p$, $\mathcal{L}_{va}^p$. If those two conditions are met, the projection vector is returned by the algorithm.

If the `find_sol` phase was able to find a solution spanning the d shares, the objective function is informative enough to allow a second (intensification) step, and the `improve_sol` phase (in Algorithm 5) is run for N_r^i iterations. At each iteration, the entire neighbourhood is constructed with the function `get_neighbours_IS`. Each neighbour results from the shift (left or right) of one window or the resizing of all the windows

Algorithm 5 Improve solution phase.

```

improve_sol_phase( $\alpha$ , TP'')
TP'' :=  $\{N_r^i, move\_steps, resize\_steps, min\_WS, max\_WS, N_n, max\_stagn\}$ 
 $\alpha_{best} = \alpha$ ;
Repeat  $N_r^i$  times
    neighbourhood = get_neighbours_IS( $\alpha$ , move_steps,
                                     resize_steps,
                                     min_WS, max_WS);

     $\alpha = select\_neighbour(@f_{obj}, \mathcal{L}_{tr}^p, N_n)$ ;
    if  $f_{obj}(\alpha, \mathcal{L}_{tr}^p) > f_{obj}(\alpha_{best}, \mathcal{L}_{tr}^p)$ 
         $\alpha_{best} = \alpha$ ;
        num_stagn = 0;
    else
        num_stagn ++;
    end

    if num_stagn > max_stagn
        return  $\alpha_{best}$ ;
    end
end
return  $\alpha_{best}$ ;
end

```

(we keep the same size for all windows). The move steps considered are given in *move_steps*, and the resize steps in *resize_steps*. The size of the windows is constrained to remain between *min_WS* and *max_WS*. The selection of the neighbour is then performed by *select_neighbour*, as a random neighbour amongst the N_n best neighbours. Using this selection strategy allows the search to avoid being trapped in local optima, ensuring a sufficient diversification. The search also memorizes the best projection obtained since the beginning of the phase in α_{best} . This is mandatory as it is allowed to select projection vectors that decrease the objective function. Eventually, the variable *num_stagn* records the number of iterations without improvement of the best solution. Once *num_stagn* is larger than *max_stagn* or when the number of iterations reaches $N_r = N_r^f + N_r^i$, the search returns the best solution α_{best} .

As far as the technical parameters are concerned, we first set the number of hops (*num_hops*) in the *find_sol* phase to allow the windows to cover all the dimensions of the traces. It enables an iteration to find

a covering set of windows when one window is incorrectly placed. Next, in the `improve_sol` phase, the more move steps (*move_steps*) and resize steps (*resize_steps*), the quicker the algorithm converges towards the optimal windows, but the longer each iteration is. We found that a good tradeoff in our context was to use *move_steps* of 1, 3 or 5 dimensions and *resize_steps* of 1 dimension. Those settings allow the iterations to be fast while still covering a large part of the search space around the solution found by the `find_sol` phase. The *min_WS* parameter typically depends on the sampling rate of the oscilloscope used in the attack: we set it to 5 which corresponds to half a cycle in our experiments, based on the intuition that dimensions next to a POI may also contain information. *max_WS* was then chosen as $2 \cdot W_{len}$, reflecting that this information can be spread on multiple clock cycles. Finally, a *max_stagn* value of 50 allows the local search to stop when it is unlikely to further improve the quality of the windows. And given the low span of the moves and the resizes, an exploration parameter N_n of 3 is enough to escape local optima and still converge towards the optimal solution.

3.4.2 Simulated experiments

We now discuss the setting of the more intuitive parameters W_{len} and T_{det} together with the performance gains obtained thanks to our specialized PP algorithm. In view of their heuristic nature, these questions are best investigated with simulated examples, where we can play with some important parameters of leaking implementations. For this purpose, we will consider a first-order masked S-box where the adversary receives N_i pairs of leakage variables of the form:

$$\begin{aligned} L_i^1 &= \text{HW}(\text{S}(x \oplus s) \oplus m) + R_i^1, \\ L_i^2 &= \text{HW}(m) + R_i^2, \end{aligned}$$

where HW is the Hamming weight function, S the AES S-box, x a plaintext byte, s a key byte, m a secret random mask, and R_i^1, R_i^2 are normally distributed noise variables with variance σ_n^2 ($1 < i \leq N_i$). For simplicity, we make sure that the N_i samples corresponding to the two shares are not overlapping. Next to these $2 \times N_i$ informative samples, we finally add $N_s - 2 \times N_i$ random samples N_j , so that N_s is the total number of samples in our simulated traces.

Setting the detection threshold. An important parameter in Algorithm 4 is the threshold value used to decide whether an improvement of the objective function is significant. In this context, a particularly convenient feature of the MMPC criteria (defined in Section 2.3.3) is

that it gradually tends to one as the number of measurements used in the detection increases. That is, given that the order of the statistical moment (e.g. $d = 2$ in our current simulations) and number of measurements used in the detection is sufficient, this criteria always reaches high values. Intuitively, it is because the MMPC relates to the statistical confidence we have in our estimated moments rather than their informativeness (see [70] for a discussion). As a result, and using such an objective function, we are able to set the detection threshold T_{det} in a completely black box manner (i.e. independent of the implementation details). Indeed, the only thing we have to guarantee is that the MMPC as computed by the objective function is significant in front of the one that would be obtained by chance, for non-informative samples. But this essentially depends on the size of the target operations. For example, the correlation between random 256-element vectors is (roughly) Gaussian-distributed⁷ with mean zero. And the probability that $\text{MMPC} > 0.2$ by chance in this case is already below the one corresponding to three σ 's (i.e. below 0.1%). Of course, one can expect slight deviations from such an ideal behaviour (e.g. so-called ghost peaks leading to non-zero mean MMPC for non-informative samples), but our next experiments will confirm that setting T_{det} to 0.2 is generally good.

Impact of W_{len} , σ_n^2 and N_i on the detection success. Given a detection threshold set as just explained, we can now evaluate the impact of different parameters on the success of our `find_sol` phase. In particular, the noise variance σ_n^2 , number of informative pairs of samples in the traces N_i and window length W_{len} are important in this respect. As just explained, we know that given a large enough number of measurements, the MMPC criteria should become larger than 0.2 for the informative samples. But it also means that if this number of measurements is not sufficient, the moments used in MCP-DPA will not be sufficiently well estimated and the detection may fail. As usual, the main parameter influencing the estimation complexity is the noise variance σ_n^2 . Yet, since we apply the objective function after projection in our PP algorithm, the size of the window W_{len} also matters here. Indeed, adding W_{len} samples with noise variance σ_n^2 implies a larger noise variance $W_{len} \times \sigma_n^2$ after projection. This is typically illustrated in the left part of Figure 3.11, where we see the impact of increasing W_{len} for two noise levels ($\sigma_n^2 = 0.1$ in the top figure, $\sigma_n^2 = 2$ in the bottom one). That is, for too large noise variances or window lengths, the estimation of the MMPC criteria is not good enough to take good decisions (i.e. is below T_{det}). In other

⁷More precise estimates can be obtained with Fisher's Z transform.

words, more measurements are needed in this case for the PP algorithm to output meaningful results. Interestingly, we also see in the right part of the figure that adding meaningful samples in the traces (i.e. increasing N_i) quite significantly mitigates the impact of large window lengths. So intuitively, traces with multiples POIs available will benefit better from our proposed method.

Time complexity. The previous results suggest that the complexity of PP algorithms is essentially a tradeoff between time and measurement complexities. That is, increasing the window length should decrease their time complexity⁸, but increase the noise after projection, and so the number of measurements needed to estimate the MMPC criteria with sufficient confidence. This is typically illustrated in Table 3.1, where we also see the benefit of having more informative samples in the traces (i.e. increasing N_i). Furthermore, Table 3.2 highlights the impact of increasing the size of the traces N_s . As in a combinatorial search, the time complexity of the PP algorithm should increase quadratically with it (more generally, it depends on N_s^d with d the number of shares in the masking scheme). Yet, increasing W_{len} or N_i can make this increase quasi-linear for some (not too large) values of N_s . Besides, note that both tables include all the constant factors related to the technical parameters in the previous section, which may affect these asymptotic predictions. Note also that these tables count the calls to the objective function for readability, but this count is not fully reflective of the PP's time complexity when changing the size of the profiling sets $\mathcal{L}_{tr}^p$ and $\mathcal{L}_{va}^p$, since larger sets also increase the complexity of each evaluation of the objective function. Yet, thanks to the parallelism of MCP-DPA attacks, the impact of these increases was limited in our experiments, leaving us with strong concrete results, as the next section will show.

3.4.3 Measured experiments

The previous simulated experiments suggest that a specialized PP algorithm can be an efficient way to find POIs in the leakage traces of masked implementations. Now, we would like to confirm this hope in front of a real case-study. For this purpose, we consider actual measurements of the first-order masked AES S-box described in Section 2.2.3. For every pair of input/output masks (m, q) , it pre-computes an S-box S^* such that $S^*(x \oplus s \oplus m) = S(x \oplus s) \oplus q$. Since this pre-computation is part of the adversary's measurements, it leads to quite memory-consuming

⁸At most linearly since the benefit of increasing the window length W_{len} saturates whenever it is not negligible in front of the number of samples in the traces N_s .

Table 3.1: Impact of W_{len} , N_i on the average number of f_{obj} calls.

$N_s = 1000$		N_i	
		5	10
W_{len}	10	7306	4681
	20	3920	3008
	30	3266	2782
	50	-	2138
	100	-	1020
	150	-	-

Table 3.2: Impact of N_s on the average number of f_{obj} calls.

	N_s		
	500	1000	2000
$W_{len} = 50, N_i = 10$	905	2138	4673

traces of $N_s = 30,000$ samples (which would be a challenging target for a combinatorial search). Furthermore, we empirically verified that our implementation does not lead to any (easy-to-detect) first-order information leakage, by running template attacks for all the time samples, and making sure that the success rate remained negligible (which should be guaranteed by the use of independent masks m and q , in order to prevent leakages based on the transitions between the the S-box input and output). Our motivation for using this setup was twofold. First, we selected a masking countermeasure based on pre-computed tables in view of the difficulty to obtain a first-order secure implementation based on other standard masking schemes such as [93] – see [6] for a recent discussion of this problem. Second, we purposely put ourselves in a challenging scenario with large traces, without trying to compress them (e.g. by reducing the sampling frequency or through educated guess). While we agree that concrete adversaries would try to exploit these possibilities, we assume that they would not always be able to compress traces up to feasible combinatorial search, and the experiments in this section aim to reflect this possibility.

We then analyzed our set of profiling and test traces, in order to evaluate the success and efficiency of our POI detection tool. We used the same MMPC criteria and detection threshold of 0.2 as previously discussed, and selected a window length W_{len} of 25, corresponding to approximately two clock cycles in our measurements: this is the only physical intuition used in our experiments. With these parameters, it

turned out that the estimation of the objective function was sufficiently accurate (for our detection threshold to make sense) with 50 profiling traces per template (i.e. 50×256 among the 500×256 measured). Based on our 1500 test traces, we then evaluated that the local search algorithm was able to return a solution within an average of 12 000 calls to f_{obj} (roughly corresponding to 7 minutes of execution time on our desktop computer). We then repeated this search multiple times in order to find several pairs of informative windows. We finally used these windows to launch multivariate (Gaussian) template attacks using 2, 4 and 8 dimensions. For this purpose, we selected the smallest windows (which turned out to contain 5 samples) and built templates for their mean values (so that each pair of window provided us with 2 dimensions). The results of these attacks are illustrated in Figure 3.12 and confirm that our tool successfully detected POIs in this challenging case. For convenience, and in order to limit our measurement needs, we estimated a 4th-order success rate which corresponds to an adversary able to enumerate 2^{32} keys. Interestingly, we see that the gain due to increased dimensionalities vanishes when moving from 4-dimension templates to 8-dimension ones. We conjecture that this mainly relates to template estimation issues. Note anyway that, as mentioned in introduction, these attacks are not aimed to be optimal from the data complexity point-of-view (since we have no guarantee to find the most informative samples). Our main goal was to provide a time-efficient POI detection tool, in a black box setting. To the best of our knowledge, previous methods for this purpose would not have been able to deal with 30,000-sample traces without an educated guess (For illustration, the product traces mentioned in footnote 1 would correspond to $900 \cdot 10^6$ samples).

3.4.4 Discussion & comparison with existing works

The previous (simulated and actual) experiments underline that the detection of POIs in the leakage traces of masked implementations is a tradeoff between the time complexity of the PPs and the amount of traces available for this purpose. In this context, it is first important to remark that the selection of a good objective function is essential. In theory, any objective function that captures higher-order statistical moments of the leakage distribution can lead to successful detections. For example, the information theoretic metric (MI) of Section 2.3.3 is a possibility. Yet, concretely, it would be more expensive to compute and estimate, so the MMPC criteria used in this section appeared more convenient for the purposes of this section (where we want to detect POIs, not to optimize their informativeness). Next, this tradeoff also draws

pretty intuitive connections between our work and naive approaches to the detection of POIs. On the one hand, (randomized) combinatorial search corresponds to one extreme scenario, where we use windows of length one, implying minimum trace requirements (so the simplest possible estimation) at the cost of maximum time complexity. On the other hand, signal integration corresponds to the other extreme scenario, where we use a single window of maximum length, implying very large trace requirements (since it implies estimating very noisy projected samples) but minimum time complexity (since the window covers the full trace). Hence, PPs can be viewed as a natural and practically useful way to explore the tradeoffs between these extremes.

Besides, we would like to end this section by discussing its differences (and complementarity) with the recent work of Reparaz et al. In summary, the results in [91] start from a different context than ours, as they primarily focus on a non-profiled attack setting. In this context, they made the useful observation that, while naive approaches to POI detection usually launch the (non-profiled) attacks for all key candidates independently, it is in fact possible to (approximately) detect these points independently of the key material first, and to launch the attacks for all the key candidates on a subset of relevant points afterwards (leading to a reduction of the time complexity by a factor roughly corresponding to the number of key hypotheses in the attack). We take the example of a masked S-box to illustrate this idea, where we manipulate a masked value $V = S(x \oplus s) \oplus M$ (with the plaintext x fixed/chosen) and its corresponding mask M . The proposal of Reparaz et al. intuitively works because the informative pairs of samples for which $\hat{I}(S; X, L(V), L(M))$ is non negligible also correspond to pairs of samples for which $\hat{I}(L(V), L(M))$ is significantly larger than 0 (for a chosen x – the CHES 2012 paper also describes known plaintext variants). Interestingly, while the first quantity can only be computed using key knowledge, the estimation of the second one does not require s to be known. So overall, this previous work does not reduce the complexity of the POI detection (which is still performed based on a combination of educated guess and combinatorial search over 800 samples in [91]). It rather makes it possible to be performed only once for all key candidates in a non-profiled setting. As a result, PP algorithms could be directly combined with this proposal, by simply using Reparaz et al.’s criteria (i.e. $\hat{I}(L(V), L(M))$) as objective function. This would extend the applicability of our work to the non-profiled setting, at the cost of a less favourable “number of measurements vs. time complexity tradeoff”. Indeed, estimating such a (non-profiled) objective function would be more

expensive than our profiled MMPC criteria. Computing such an objective function implies (constant but significant) performance overheads, because it requires applying Bayes’ law and marginalizing over the key hypotheses, as the information theoretic metric.

This discussion eventually allows us to clarify what we mean by “black box evaluation”. In our experiments, we did not require implementation details nor mask knowledge, but assumed key knowledge to speed up estimations. We believe this realistically captures the constraints of an evaluation lab. But extending our work to a non-profiled setting may be achieved by changing the objective function, as just discussed. In this context, it is important to remark that the POI detection not only depends on the tradeoff between number of measurements and time complexity, but also on the “engineering intuition available” and “type of information detected”. Starting with the engineering intuition, we can observe that directly using $\hat{I}(L(V), L(M))$ as objective function would make the setting of the detection threshold more challenging (i.e. require some engineering intuition). Indeed, the asymptotic value of this objective function depends on the leakage function, in contrast with the MMPC criteria that asymptotically tends to be independent of the leakage function (which allows a well motivated and black box choice of the detection threshold, as discussed in Section 3.4.2). Note that using the confidence level of an hypothesis test is an interesting alternative for this purpose, and will be investigated in the next section. Next, the type of information detected is another parameter that may allow speeding up POI detection. Namely, one of the reasons making an objective function costly to estimate is that its argument is an intermediate computation of which the output typically takes 256 possible values. This means that we want to learn information about a random variable with (relatively) large input range. In this respect, a natural solution is to consider a target with smaller range, and at the extreme, a binary random variable. This is in fact what is advertised in the *fixed vs. random* leakage detection test discussed in Section 3.1. For this test, the authors try to detect leakage points where there is a significant difference between two classes: the first corresponding to fixed plaintexts, the second corresponding to random plaintexts. On the positive side, this allows detecting leaking points without any profiling, with a simple (binary) hypothesis test, which indeed provides easy-to-compute confidence levels. On the negative side, there is no guarantee that the detected samples all correspond to useful (key-dependent) information, nor that all POIs can be detected in this way. In practice though, such tests have been shown quite powerful for detecting POIs in concrete implementations of, e.g. the AES. Note that

leakage detection tests have mainly been applied to unprotected implementations so far, but their extension to masking is conceptually simple (see, e.g. [6]).

Summarizing, this work puts forward the interest of PP algorithms in the context of profiled black box evaluations, where they allow detecting leakage points which are certainly “of interest” (since they are key-dependent), with reasonable measurement vs. estimation cost and no implementation details. Many variations are possible in order to move to a non-profiled context, with a price to pay in estimation cost and/or need of implementation details and/or type of information detected. But all these changes of conditions will only be reflected by a change of objective function, hence preserve the relevance of our main contribution (i.e. the specialized PP algorithm). In this respect, it is important to note that independently of the evaluation context and objective function, trading time complexity for more measurements anyway becomes necessary at some point, when traces become too large for exhaustive combinatorial search. Last but not least, we recall that despite black box in the sense described in this section, our techniques still rely on some assumptions (e.g. whether the leakages are value-based or transition-based [6]). Using non-specific tests such as [38, 66] can mitigate this requirement (although it does not totally suppress it).

Before concluding, let us recall that in all these cases, the POI detection is heuristic (which is especially clear in our algorithms taking advantage of local search). Therefore, they do not guarantee that the best (i.e. most informative) POIs are found. An exhaustive analysis remains the only option for this purpose.

3.4.5 Conclusion

In this work we proposed an efficient method for finding POIs in the leakage traces of cryptographic implementations. We exploited a combination of PP and local search for this purpose, and discussed how to adapt it to the side-channel cryptanalysis problem. One of the main advantages of the method is its genericity, as it can be applied to any implementation, by simply adapting its objective function. Besides, it has very low memory requirements compared to state-of-the-art solutions and (although heuristic) works in practical time complexity. We applied our basic and specialized PP algorithms to two case studies of unprotected and 2-share masked implementations to validate our claims. Extending the specialized version to more shares would be straightforward, since this number of shares (i.e. d) is a parameter in our search algorithms.

Among the interesting open problems, we believe investigating the informativeness of the projected samples obtained with PPs in the context of protected implementations is promising – it was essentially left out of our analysis so far. Different approaches could be considered for this purpose. One would be to refine the projection vectors, possibly based on an information theoretic objective function that would better reflect the data complexity of the resulting attacks. An alternative one would be to exploit non-linear projections, e.g. inspired by the “product combining”, frequently used in second-order DPA [83, 102]. Yet, preliminary results suggest that non-linear projections may be hard(er) to exploit because the addition of non-informative samples when computing the objective function has higher impact on the (non-Gaussian) noise in this case. Besides, testing new objective functions that are cheap to compute and estimate, in the profiled and non-profiled settings, is another interesting research direction.

3.5 From leakage detection to POI detection

The conclusions drawn in Section 3.1.2 lead to the next contribution of the work reported in this thesis. That is, when extending leakage detection towards the detection of POIs, the ρ -test proposed in Section 3.1.1 naturally gains additional interest, since it provides more intuitions regarding the exploitable samples in side-channel traces. More precisely, it allows a better selection of POIs based on the criteria that these POIs depend on an enumerable part of the key. It also maximizes the SNR metric that can be easily connected to the worst-case complexity of standard DPA attacks [29]. Therefore, and most importantly, our results directly imply that the automated tools for the detection of POIs proposed in Section 3.4 are also applicable in a full black box setting, without any key knowledge, by simply adapting the objective function used in their optimization (i.e. replacing it by the ρ -test in this section). We finally confirm this claim by an additional experimental evaluation, in the context of a first-order secure masked implementation. We also put forward that the detection of a threshold for which an improvement of the objective function is considered as significative in the optimizations of Section 3.4 is made easier when using the ρ -test (and suggest a minor improvement of these last methods, by taking advantage of cross-validation when evaluating this objective function).

3.5.1 The ρ -test as a POI detection tool

One important conclusion of the experiments of Section 3.1.2 is that leakage detection based on a ρ -test provides useful intuitions regarding the exploitable samples in side-channel traces. As a result, it is a good candidate for the more specific task of detecting POIs for mounting an attack. In this section, we conclude the chapter by putting forward that a ρ -test is in fact perfectly suited for integration in (and improvement of) the POI detection tool proposed in Section 3.4. For this purpose, we describe our improvements based on our proposed ρ -test, and provide some experimental confirmation of our claims.

Note that in general, the problem of detecting POIs is relatively easy in the context of unprotected implementations. Indeed, exhaustive analysis is usually feasible in this case, and it is even possible to look for optimal transforms that project the samples towards small (hence easier-to-evaluate) subspaces such that most of their informativeness is preserved, e.g. using Principal Component Analysis (PCA) [2], which maximizes the side-channel signal, or Linear Discriminant Analysis (LDA) [103], which maximizes the side-channel SNR. In fact, in this context, any criteria can be easily optimized using local search (Section 3.3 and [71]), and most criteria are essentially equivalent anyway (i.e. correlation, SNR, mutual information and success rate [29, 65]). Therefore, our focus will be on the more challenging case of masked implementation, which requires the specialized local search proposed in Section 3.4.

This contribution

We first recall that the POI detection tool of Section 3.4 is *black box* in the sense that it does not require any knowledge of the target implementation. By contrast, it does require *key profiling*, since the MCP-DPA distinguisher is a profiled one. In this respect, our first contribution is the simple but useful observation that one can easily apply such a black box POI detection without key profiling, by simply profiling the MCP-DPA objective function based on plaintext knowledge, just as the ρ -test in this chapter. Indeed, when detecting POIs, it is sufficient to know the leakage model up to a permutation corresponding to key knowledge (a quite similar idea is exploited in [91] for similar purposes). As discussed in Section 3.1.2, this solution will suffer from the (minor) risk of detecting plaintext samples, but as will be detailed next, this can be easily mitigated in practice.

Based on these premises, our second contribution is the equally sim-

ple observation that our ρ -test can be used identically with the MCP-DPA distinguisher, so is theoretically eligible for detecting leakages and POIs of any order. And this observation leads to our third contribution, namely that by replacing the MCP-DPA objective function by the ρ -test (based on CPA or MCP-DPA), we obtain a very simple and rigorous way to set the detection threshold in Algorithm 3. That is, one just has to use the same “five sigma rule” as used in the leakage detections of Figures 3.2 and 3.3. Note that by changing the objective function and selection of a detection threshold in this way, we benefit from the additional advantage of estimating the objective function with cross-validation, which is another (minor) improvement over the previous method.

3.5.2 Experimental validation

In order to confirm the previous claims, we tested Algorithm 3 using exactly the previously described modifications, based on the same target implementation and measurement setup (Section 3.4.3). This masked implementation which leads to large traces with $N_s = 30,000$ samples (for which an exhaustive analysis of all the pairs of samples is out of reach). We verified that the implementation does not lead to any first-order leakages with the ρ -based test. We further set the window length to 25 samples, which corresponds to a bit more than two clock cycles at our clock frequency and sampling rate. With these parameters, the local search was able to return a solution within the same number of objective function calls, namely $\approx 12\,000$ on average. An example of leakage trace together with windows obtained thanks to Algorithm 3 is given in Figure 3.13. As clear from the zoomed plots at the bottom of the figure, the selection of POIs corresponds to leakage samples that combine the precomputation and masked S-box computation. Interestingly, we could expect some false positives due to the detection of plaintext bytes that is possible in our non-profiled scenario. However, the `improve_solution` of Algorithm 3 (where the window size is adapted to be most informative) combined with the fact that the most informative leakage samples in our traces correspond to memory accesses (i.e. the S-box computations) prevented these to happen. Note that even if the leakage of the plaintext manipulations was more informative, we could easily “mark” the cycles that correspond to plaintext knowledge only, and exclude them from our optimization. Since the number of POIs corresponding to a single plaintext byte is usually limited, this would lead to the detection of a valid pair of POIs after a couple of iterations of Algorithm 3. Besides, we note that Simple Power Analysis, or DPA against the plaintext (before

it is XORed with the masks) are other simple ways to gain the minimum intuition about the time localization of POIs to make non-profiled local search applicable.

3.5.3 Conclusion

To conclude, we insist that this last section has admittedly limited technical novelty. However, we believe that the connections made are important to raise awareness that up to the selection of POIs in the leakage trace, side-channel security evaluations can essentially be performed in a black box way, and without any key profiling. In this respect, leakage detection and the detection of POIs are indeed very related tasks, with the significant difference that the latter has to take the exploitability of the detected samples into account. And this is exactly the difference between simple t-tests and (slightly) more measurement-intensive ρ -tests based on larger leakage partitions. Note that the non-profiled detection in this section only applies to the first/last block cipher rounds (i.e. before diffusion is complete), which captures many relevant practical scenarios but could be an issue, e.g. in contexts where these extreme rounds are better protected than the central ones. Besides, and more generally, we recall that as soon as the POIs are detected and the evaluator has to build a model for these samples, key profiling becomes strictly necessary if the worst-case security level has to be evaluated [117].

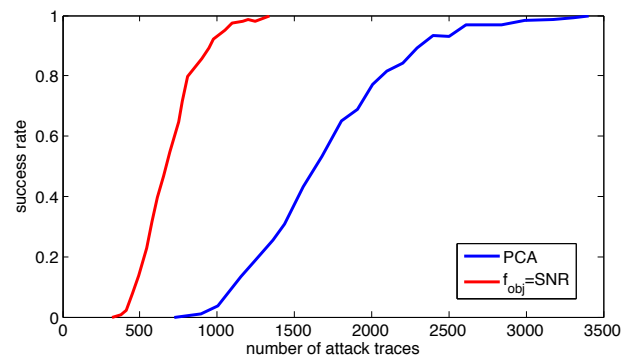


Figure 3.10: Template attack success rates from the DPA contest traces.

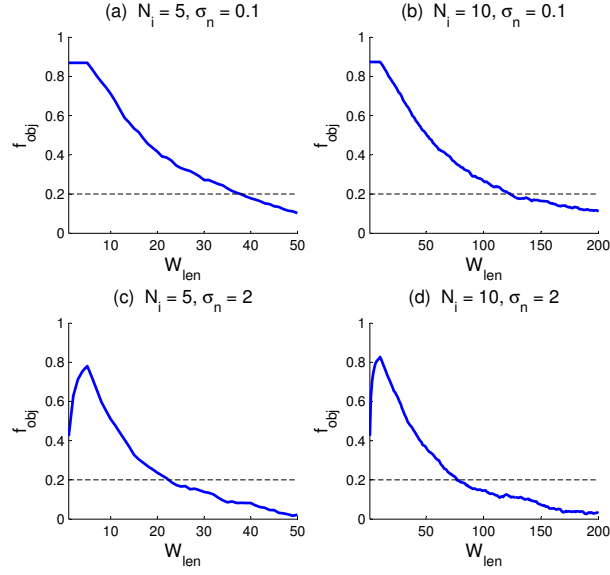


Figure 3.11: Incidence of the window length W_{len} on the information detection.

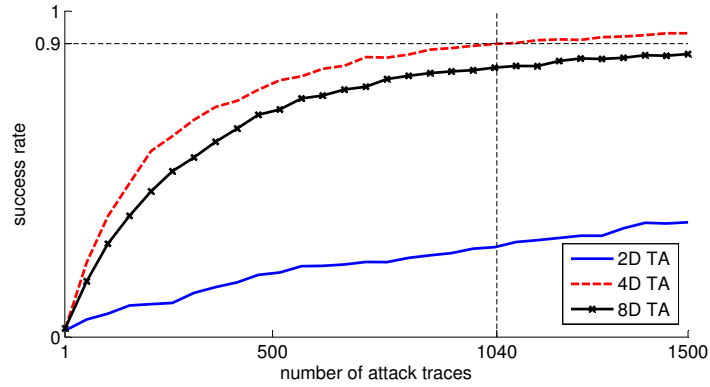


Figure 3.12: 4th-order success rates of multivariate template attacks.

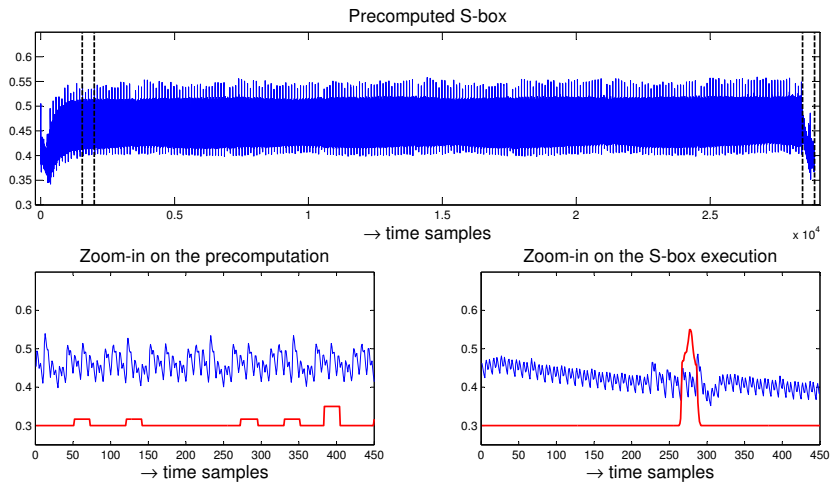


Figure 3.13: Non-profiled detection of POIs based on the ρ -test.

Chapter 4

Leakage certification

Side-channel attacks are an important threat against the security of modern embedded devices. As a result, the search for efficient approaches to secure cryptographic implementations against such attacks has been an ongoing process over the last 15 years. Sound tools for quantifying physical leakages are a central ingredient for this purpose, since they are necessary to balance the implementation cost of concrete countermeasures with the security improvements they provide. Hence, while early countermeasures came with proposals of security evaluations that were sometimes specialized to the countermeasure, more recent work has investigated the possibility to consider evaluation methods that generally apply to any countermeasure. The unified evaluation framework proposed at Eurocrypt 2009 (Section 2.2.4) is a typical attempt in this direction. It suggests analysing cryptographic implementations with a combination of information theoretic and security metrics. The first ones aim at measuring the (worst-case) information leakage independently of its exploitation by the adversary, and are typically instantiated with the Mutual Information (MI). The second ones aim at quantifying how efficiently an adversary can take advantage of this leakage in order to turn it into (e.g.) a key recovery, and are typically instantiated with a success rate.

In this context, an important observation is that most side-channel attacks, and in particular any standard DPA attack, require a leakage model [65]. This model usually corresponds to an estimation of the leakage Probability Density Function (PDF), possibly simplified to certain statistical moments. Since the exact distribution of (e.g.) power consumption or electromagnetic radiation measurements is generally unknown, it raises the problem that any physical security evaluation is possibly biased by model errors. In other words, security evaluations

ideally require a perfect leakage model (so that all the information is extracted from the measurements). But in practice models are never perfect, and thus the quality of the evaluation may highly depend on the quality of the evaluator. This intuition can be captured with the notion of Perceived Information (PI – described in Section 2.3.3) - that is nothing else than an estimation of the MI biased by the side-channel evaluator’s model [90]. Namely, the MI captures the worst-case security level of an implementation, as it corresponds to an (hypothetical) adversary who can perfectly profile the leakage PDF. By contrast, the PI captures its practical counterpart where actual (statistical) estimation procedures are used by an evaluator, in order to profile the leakage PDF. The previous formal tools provide a sound basis for discussing the evaluation question “*how good is my leakage model?*”. The answer to this question actually corresponds to the difference between the MI and the PI. Nevertheless, we remain with the problem that the MI is generally unknown (just as the actual leakage PDF), which makes it impossible to compute this difference directly.

Picking up on this problem, the first section introduces a first “leakage certification” method. Intuitively, leakage certification starts from the fact that actual leakage models are obtained via PDF estimation, which may lead to both estimation and assumption errors. As a result, and since it seems hard to enforce that such estimated models are perfect, the best that one can hope is to guarantee that they are “good enough”. For estimation errors, this is easily verified using standard cross-validation techniques (in general, estimation errors can anyway be made arbitrarily small by measuring more). For assumption errors, things are more difficult since they require us to find out whether the estimated model is close to an (unknown) perfect model. Interestingly, we show that indirect approaches allow determining if this condition is respected, essentially by comparing the model errors caused by incorrect assumptions to estimation errors.

Yet, this certification method involves a number of technical ingredients. That is, the evaluator has to characterize the leakages of the chip and his model with distance cumulative distributions. Although working with distances is a sound approach to compare distributions and does not require any assumption on the leakage distribution, it comes at the cost of quite computationally intensive tools. In a second section of this chapter, we propose a solution in order to mitigate the latter drawback. We show that a statistical moment based comparison can provide a computationally cheaper and conceptually simpler certification procedure but obtained at the cost of mild assumptions on the statistical

distributions in hand.

Related publications

1. François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. How to certify the leakage of a chip? In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 459–476. Springer, 2014.
2. François Durvaux and François-Xavier Standaert. Towards easy leakage certification. Currently submitted, 2015.

4.1 How to certify the leakage of a chip?

In this context, we start with the preliminary observation that understanding these fair evaluation issues requires clearly distinguishing between estimation errors and assumption errors, leading to three main contributions. First, we show how the cross-validation, described in Section 2.3.4, can be used in order to precisely gauge the convergence of an estimated model. Doing so, we put forward that certain evaluation metrics (e.g. Pearson’s correlation or PI) are better suited for this purpose. Second, we propose a method for measuring assumption errors in side-channel attacks, taking advantage of the distance sampling technique introduced in [114]. We argue that it allows detecting imperfect hypotheses without any knowledge of the true leakage distribution¹! Third, we combine these tools in order to determine the probability that a model error is due to estimation or assumption issues. We then discuss the (im)possibility to precisely (and generally) bound the resulting information loss. We also provide pragmatic guidelines for physical security evaluators. For illustration, we apply these contributions to actual measurements obtained from an AES implementation in an embedded microcontroller (setup described in Section 2.3.1). As a result and for the first time, we are able to certify that the leakage of a chip (i.e. its worst-case security level) is close to the one we are able to extract.

¹By contrast, the direct solution for quantifying the PI/MI distance would be to compute a statistical (e.g. Kullback-Leibler) distance between the adversary’s model and the actual leakages. But it requires knowing the true leakage distribution.

These results have implications for the certification of any cryptographic product against side-channel attacks - as they provide solutions to guarantee that the evaluation made by laboratories is based in sound assumptions. They could also be used to improve the comparison of measurement setups such as envisioned by the DPA contest v3 [76]. Namely, this contest suggests comparing the quality of side-channel measurements with a CPA based on an a-priori leakage model. But this implies that the best traces are those that best comply with this a-priori, independently of their true informativeness. Using the PI to compare the setups would already allow each participant to choose his leakage assumptions. And using the cross-validation and distance sampling techniques described in this work would allow determining how relevant these assumptions are.

4.1.1 Estimation errors and cross-validation

As a starting point, we represented illustrative traces corresponding to our measurement setup of Section 2.3.1 in Figure 4.1. Considering an 8-bit implementation allows us to make the implicit assumption that there's not interaction between the bytes that cannot be captured by the model. The figure further contains the SNRs and correlation coefficients of a CPA using Hamming weight leakage model and targeting the S-box output. While insufficient for fair security evaluations as stated below, these metrics are interesting preliminary steps since they indicate the parts of the traces where useful information lies, i.e. the points-of-interest. In the following, we extract a number of illustrative figures from meaningful samples.

From a methodological point of view, the impact of cross-validation is best represented with the box plot of Figure 4.2: it contains the PI of point 2605 in the traces, estimated with Gaussian templates and a stochastic model using a 17-element linear basis for the bits of the S-box input and output. This point is the most informative one in our experiments (across all the measurements and estimation procedures we tried). Results show that the PI estimated with Gaussian templates is higher - hence suggesting that the basis used in our regression-based profiling was not fully reflective of the chip activity for this sample. More importantly, we observe that the estimation converges quickly (as the spread of our 10 PI estimates decreases quickly with the number of traces). As expected, this convergence is faster for regression-based profiling, reflecting the smaller number of parameters to estimate in this case. Note that we also performed this cross-validation for the Kernel-based PDF estimation described in Section 2.3.2. The results are

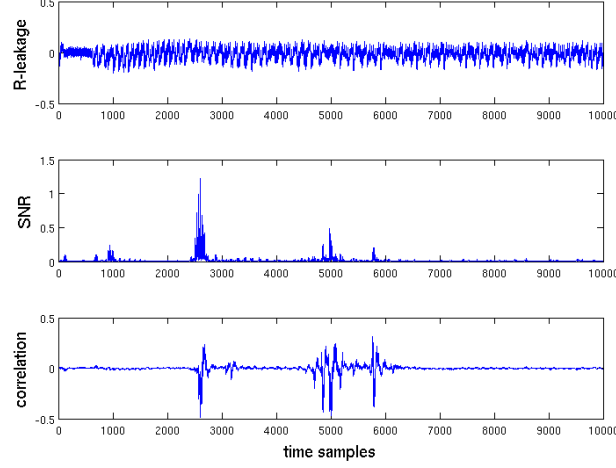


Figure 4.1: Measurements of a key addition and S-box layer in the first round of the AES.

reported in Figure 4.3. Even though the Kernel-based PDF estimation is slower to converge, both the expected value of the PI and its spread suggest that these two density estimation techniques provide equally satisfying results in our implementation context.

A natural next step is to analyse the quantity of information given by alternative leakage points. An example is given in Figure 4.4 (where we only plot the expected value of the PI). The left part of the figure exactly corresponds to the most informative point of Figure 4.2. The right part of the figure is computed with a later sample (time 4978) that (we assumed) corresponds to the computation of the S-box output. Interestingly, we observe that while this second point is less informative, it is more accurately explained by a stochastic model using the S-box output bits as a basis, hence confirming our expectations. Eventually, we also investigated the additional information gathered when performing multivariate attacks in Figure 4.5. For this purpose, we measured an additional set of traces by replacing the resistor of Section 2.3.1 by a $2\ \mu H$ inductance (everything else remains the same). On the left-hand side of the figure, we considered a couple of points (2605 and 4978) coming from the original setup. On the right-hand side of the figure, we considered a single point (2605) coming from the two different setups. This experiment clearly suggests that combining information from different

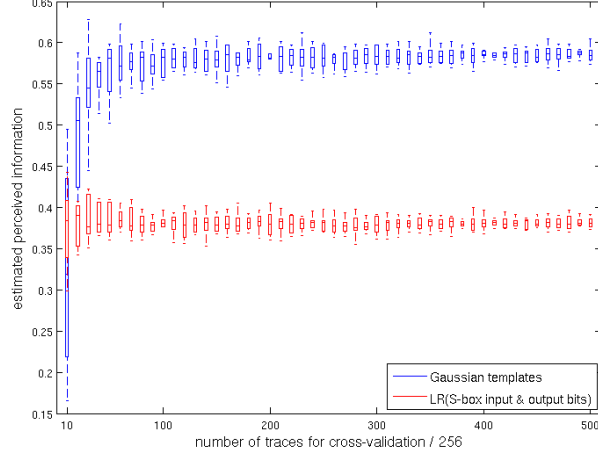


Figure 4.2: Perceived information estimated from Gaussian templates and LR-based models, with cross-validation (target point 2605 from the measurements).

operations leads to more PI than combining information from different setups. It naturally fits with the intuition that two different block cipher operations (corresponding to different intermediate values) lead to more information leakage (i.e. less correlation) than the same operation measured with two different (yet similar) measurement setups. Many variations of such evaluations are possible (for more samples, estimation procedures, ...). For simplicity, we will limit our discussion to the previous examples, and use them to further discuss the critical question of assumption errors in the next section.

4.1.2 Assumption errors and distance sampling

Looking at Figures 4.2 and 4.4, we can conclude that our estimation of the PI is reasonably accurate and that Gaussian templates are able to extract a given amount of information from the measurements. Nevertheless, such pictures still do not provide any clue about the closeness between our estimated PI and the (true, unknown) MI. As previously mentioned in introduction, evaluating the deviation between the PI and MI is generally hard. In theory, the standard approach for evaluating such a deviation would be to compute a statistical (e.g. Kullback-Leibler) distance $\hat{D}_{KL}(\hat{Pr}_{\text{model}}, Pr_{\text{chip}})$. But this requires knowing the (unknown)

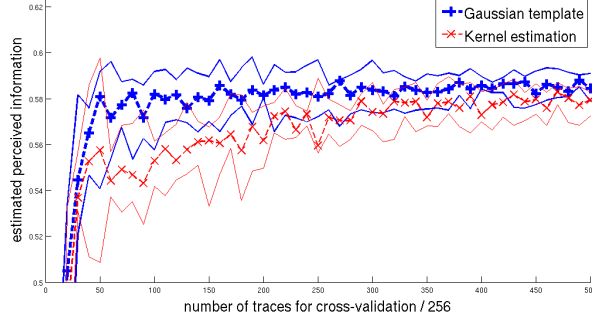


Figure 4.3: Perceived information quantiles estimated from Gaussian templates and Kernels, with cross-validation (target point 2605).

distribution Pr_{chip} , leading to an obvious chicken and egg problem.

Since standard probabilistic distances cannot be computed, an alternative solution that we will apply is to confront the test samples output by the device with estimated samples produced with the evaluator’s model. In order to check their coherence, we essentially need a goodness-of-fit test. While several such tests exist in the literature for unidimensional distributions (e.g. Kolmogorov–Smirnov [14] or Cramér–von–Mises [1]), much fewer solutions exist that generalize to multivariate statistics. Since we additionally need a test that applies to any distribution, possibly dealing with correlated leakage points, a natural proposal is to exploit statistics based on spacings (or interpoint distance) [86]. The basic idea of such a test is to reduce the dimensionality of the problem by comparing the distribution of distances between pairs of points, consequently simplifying it into a one-dimensional goodness-of-fit test again. It exploits the fact that two multidimensional distributions $\mathcal{F}$ and $\mathcal{G}$ are equal if and only if the variables $\mathbf{X} \sim \mathcal{F}$ and $\mathbf{Y} \sim \mathcal{G}$ generate identical distributions for the distances $D(\mathbf{X}_1, \mathbf{X}_2)$, $D(\mathbf{Y}_1, \mathbf{Y}_2)$ and $D(\mathbf{X}_3, \mathbf{Y}_3)$ [9, 57]. In our evaluation context, we can simply check if the distance between pairs of simulated samples (generated with a profiled model) and the distance between simulated and actual samples behave differently. If the model estimated during the profiling phase of a side-channel attack is accurate, then the distance distributions should be close. Otherwise, there will be a discrepancy that the test will be able to detect, as we now detail.

The first step of our test for the detection of incorrect assumptions

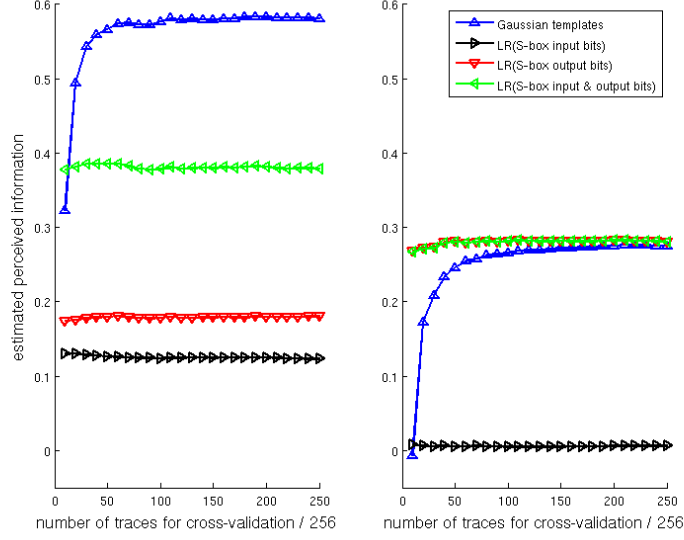


Figure 4.4: PI for different PDF estimation techniques and two leakage points. Left: most informative one (2605), right: other point of interest (4978).

is to compute the simulated distance cumulative distribution as follows:

$$f_{\text{sim}}(d, s, x) = \Pr \left[L_y^1 - L_y^2 \leq d \mid L_y^1, L_y^2 \sim \hat{\text{Pr}}_{\text{model}}[L_y | s, x] \right].$$

Since the evaluator has an analytical expression for $\hat{\text{Pr}}_{\text{model}}$, this cumulative distribution is easily obtained. Next, we compute the sampled distance cumulative distribution from the test sample set $\mathcal{L}_Y^t$ as follows:

$$\hat{g}_{N_t}(d, s, x) = \Pr \left[l_y^i - l_y^j \leq d \mid \{l_y^i\}_{1 \leq i \leq N_t} \sim \hat{\text{Pr}}_{\text{model}}[L_y | s, x], \{l_y^j\}_{1 \leq j \leq N_t} = \mathcal{L}_Y^t \right].$$

Eventually, we need to detect how similar f_{sim} and $\hat{g}_{N_t}$ are, which is made easy since these cumulative distributions are now univariate. Hence, we can compute the distance between them by estimating the Cramér–von–Mises divergence:

$$\text{CvM}(f_{\text{sim}}, \hat{g}_{N_t}) = \int_{-\infty}^{\infty} [f_{\text{sim}}(x) - \hat{g}_{N_t}(x)]^2 dx.$$

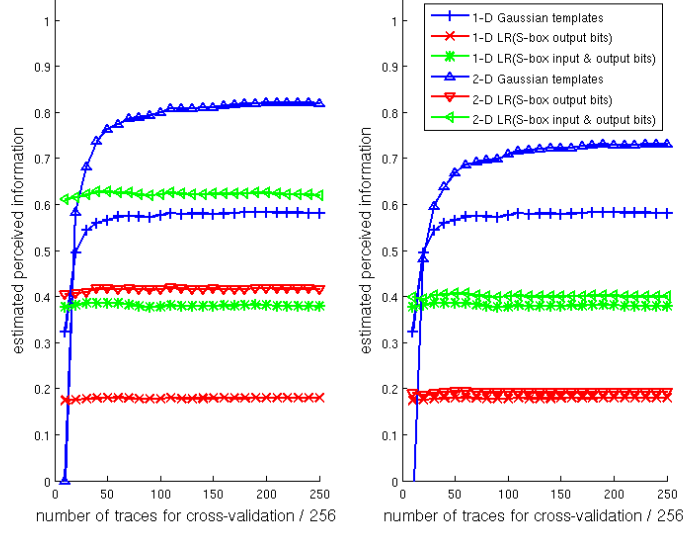


Figure 4.5: PI for univariate and multivariate leakage models. Left: two points (2605, 4978) coming from the original measurements. Right: multi-channel attack exploiting the same point (2605) from resistor- and inductance-based measurements.

As the number of samples in the estimation increases, this divergence should gradually tend towards zero provided the model assumptions are correct.

4.1.3 Experimental results

As in the previous section, we applied cross-validation in order to compute the Cramér-von-Mises divergence between the distance distributions. That is, for each of the 256 target intermediate values, we generated 10 different estimates $\hat{g}_{N_t}^{(j)}(d, s, x)$ and computed $\text{CvM}^{(j)}(f_{\text{sim}}, \hat{g}_{N_t})$ from them. An exemplary evaluation is given in Figure 4.6 for the same leakage point and estimation methods as in Figure 4.2. For simplicity, we plotted a picture containing the 256 (average) estimates at once². It shows that Gaussian templates better converge towards a small divergence of the distance distributions. It is also noticeable that regression-based models lead to more outliers, corresponding to values y for which

²It is also possible to investigate the quality of the model for any given $y = x \oplus s$.

the leakage L_y is better approximated. Figure 4.7 additionally provides the quantiles of the Cramér–von–Mises divergence for both univariate and bivariate distributions (i.e. corresponding to the PIs in Figure 4.5, original setup only). Interestingly, we observe that the better accuracy of Gaussian templates compared to regression-based models decreases when considering the second leakage point. This perfectly fits the intuition that we add a dimension that is better explained by a linear basis (as it corresponds to the right point in Figure 4.4). Note that any incorrect assumption would eventually lead the CvM divergence to saturate.

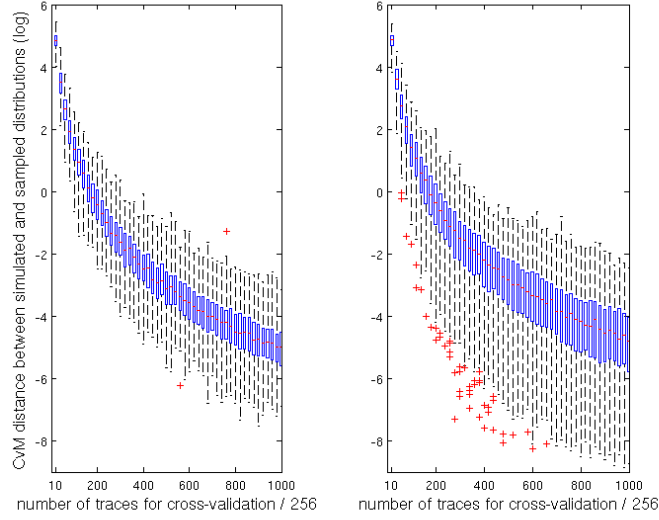


Figure 4.6: Cramér–von–Mises divergence between simulated and sampled distributions, with cross-validation (target point 2605). Left: Gaussian templates, right: LR-based estimation (S-box input and output bits).

4.1.4 Estimation vs. assumption errors

From an evaluator’s point of view, assumption errors are naturally the most damaging (since estimation errors can be made arbitrarily small by measuring more). In this respect, an important problem that we answer in this section is to determine whether a model error comes from estimation or assumption issues. For this purpose, the first statistic we need to

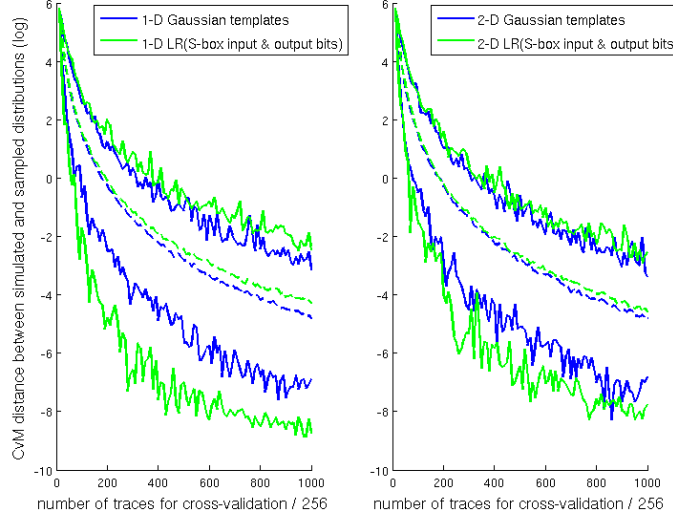


Figure 4.7: Median, min and max of the CvM divergence btw. simulated and sampled distributions for Gaussian templates and LR-based models. Left: univariate attack (sample 2605), right: bivariate attack (samples 2605 and 4978).

evaluate is the *sampled* simulated distance cumulative distribution (for a given number of test traces N_t). This is the estimated counterpart of the distribution f_{sim} defined in Section 4.1.2:

$$\hat{f}_{\text{sim}}^{N_t}(d, s, x) = \Pr \left[l_y^i - l_y^j \leq d \mid \{l_y^i, l_y^j\}_{1 \leq i \neq j \leq N_t} \sim \hat{\text{Pr}}_{\text{model}}[L_y | s, x] \right].$$

From this definition, our main interest is to know, for a given divergence between f_{sim} and $\hat{f}_{\text{sim}}^{N_t}$, what is the probability for this divergence to be observed for the chosen amount of test traces N_t . This probability is directly given by the following cumulative divergence distribution:

$$\hat{\text{Div}}_{N_t}(x) = \Pr \left[\hat{\text{CvM}}(f_{\text{sim}}, \hat{f}_{\text{sim}}^{N_t}) \leq x \right].$$

How to exploit this distribution is then illustrated in Figure 4.8. For each model $\hat{\text{Pr}}_{\text{model}}^{(j)}$ estimated during cross-validation, we build the corresponding $\hat{\text{Div}}_{N_t}^{(j)}$'s (i.e. the cumulative distributions in the figure). The cross-validation additionally provides (for each cumulative distribution) a value for $\hat{\text{CvM}}^{(j)}(f_{\text{sim}}, \hat{g}_{N_t})$ estimated from the actual leakage samples

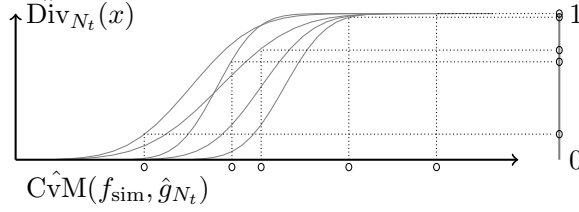


Figure 4.8: Model divergence estimation.

in the test set: they correspond to the small circles below the X axis in the figure. Eventually, we just derive:

$$\hat{\text{Div}}_{N_t}^{(j)} \left(\text{C}\hat{\text{v}}\text{M}^{(j)}(f_{\text{sim}}, \hat{g}_{N_t}) \right).$$

Computing this statistic is simply obtained by projecting the circles towards the Y axis in the figure. Large values indicate that there is a small probability for the observed samples to follow the simulated distributions. That is, the hypothesis, which states that the model is correct, is therefore rejected. More precisely, high $\hat{\text{Div}}_{N_t}^{(j)}$ correspond to low p -values (i.e. the probability that the null hypothesis holds) with $p^{(j)} = 1 - \hat{\text{Div}}_{N_t}^{(j)}$. Thanks to cross-validation, we can obtain 10 such values, leading to answers laid on a $[0; 1]$ interval, indicating the accuracy of each estimated model. Values of $\hat{\text{Div}}_{N_t}^{(j)}$ (resp. p -values $p^{(j)}$) that are grouped towards the top (resp. bottom) of the interval indicate that the assumptions used to estimate these models are probably incorrect. We will further focus on the p -values.

An illustration of this method is given in Figure 4.9 for different Gaussian templates and regression-based profiling efforts, in function of the number of traces in the cross-validation set. It clearly exhibits that as this number of traces increases (hence, the estimation errors decrease), the regression approach suffers from assumption errors with high probability. Actually, the intermediate values for which these errors occur first are the ones already detected in the previous section, for which the leakage variable L_y cannot be precisely approximated given our choice of basis. By contrast, no such errors are detected for the Gaussian templates (up to the amount of traces measured in our experiments). This process can be further systematized to all intermediate values, as in Figure 4.10. It allows an evaluator to determine the number of measurements necessary for the assumption errors to become significant in front of estimation ones.

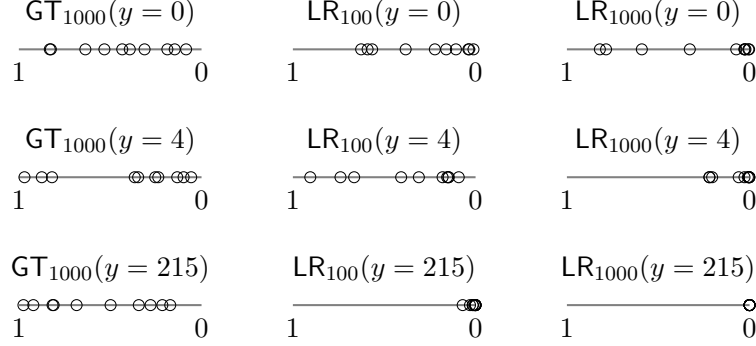


Figure 4.9: Probability that the model is correct (i.e. p -values of a “no assumption error” hypothesis) for Gaussian templates (GT) and regression-based models (LR) corresponding to different target intermediate values y , in function of N_t (in subscript), on sample 2605.

4.1.5 Can we bound the information loss?

Interestingly, most assumptions will eventually be detected as incorrect when the number of traces in a side-channel evaluation increases³. As detailed in introduction, it directly raises the question whether the information loss due to such assumption errors can be bounded? Intuitively, the “threshold” value N_t^{th} for which they are detected by our test provides a measure of their “amplitude” (since errors that are detected earlier should be larger in some sense). In this section, we discuss whether this intuition can be exploited quantitatively. Note that the following reasoning is intentionally more informal than the previous technical contributions: it mainly aims at explaining why measuring the MI-PI difference in general (i.e. independent of the leakage distribution) is hard.

Ideal expectation. Taking the example of Figure 4.10, we see that the stochastic model is already detected as imperfect after (roughly) 100×256 traces in the cross-validation set. This means that at this point in our evaluations, the assumption errors start to be significant in front of the estimation errors. As a result, a natural idea to quantify

³Non-parametric PDF estimation methods (e.g. as described in Section 2.3.2) could be viewed as an exception to this fact, assuming that the sets of profiling traces $\mathcal{L}_Y^p$ and test traces $\mathcal{L}_Y^t$ come from the same distribution. Yet, this assumption may turn out to be contradicted in practice because of technological mismatches [33, 90], in which case the detection of assumption errors remains critical even with such tools.

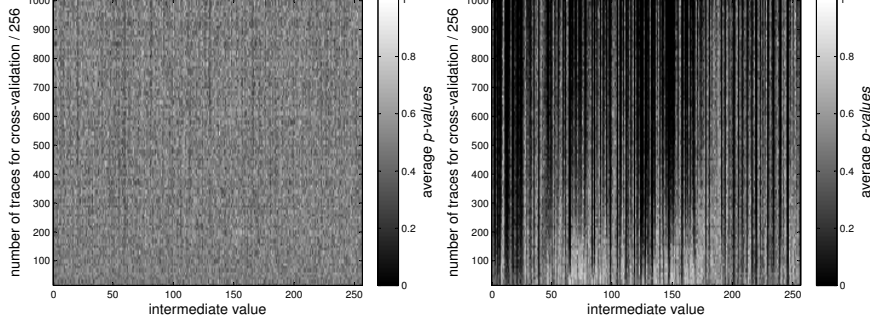


Figure 4.10: Average p -values for Gaussian templates (left) and regression-based models with a 17-element basis (right) corresponding to all the target intermediate values y , in function of N_t , on sample 2605

the information loss due to assumption errors is to compute the (easier to evaluate) information loss due to estimation errors occurring for this number of test traces. For this purpose, first note that the PI computations in Section 4.1.1 are done globally (i.e. based on 10 estimations from $256 \times N_t$ traces). By contrast, the test in Section 4.1.2 is performed per intermediate value (i.e. based on 256×10 estimations from N_t traces). So in order to be comparable, we must repeat the estimation of the PI based on 256×10 estimations from N_t test traces as well. Next, we can measure the estimation error in function of the number of traces in the cross-validation set, e.g. by computing the square root of the Mean Square Error (MSE) for the PI based on these new estimates. The result of this experiment is given in the left part of Figure 4.11 (as an alternative, we can also derive the quantiles of the estimated PI as in Figure 4.12). After 100 traces, the root MSE approximately equals 0.29 for the regression-based profiling. Our ideal expectation is that by adding up the estimated PI obtained for LR-based profiling in Section 4.1.1 (≈ 0.38) with this value, we would obtain a “bound” on the MI of ≈ 0.67 . Of course, this bound would be probabilistic (i.e. with a certain chance that the information loss exceeds 0.29). A more confident bound would then be obtained by summing this root MSE twice, leading to an hypothetical MI of 0.96. Assuming the PI estimates to be Gaussian-distributed, the probability that the MI exceeds these bounds could also be quantified (to 31.8% and 4.6%, i.e. corresponding to the addition of one or two standard deviations to the mean of a normal distribution). The quantiles in Figure 4.12 lead to similar results (confirming the Gaussian assumption).

Interpretation. Such an information theoretic evaluation would conclude that for leakage sample 2605 of our resistor-based measurements: (1) we identified a regression-based attack able to exploit a PI of ≈ 0.38 ; (2) we identified a template attack able to exploit a PI of ≈ 0.58 ; (3) with some confidence, there may exist a stronger attack able to exploit an hypothetical MI of ≈ 0.67 (that would be identified with better assumptions); (4) with higher confidence, there may exist an even stronger attack able to exploit an hypothetical MI of ≈ 0.96 . Tighter “bounds” can be obtained by using our Gaussian templates, since we could not detect any assumption error in this case. That is, we would then expect that the assumption errors will be lower than the estimation errors on the PI with 256 000 traces in the cross-validation set. This would lead to MI bounds of $\approx 0.58 + 0.11 = 0.69$ (with some confidence) and $\approx 0.58 + 2 \times 0.11 = 0.8$ (with higher confidence). As mentioned in Section 2.3.3, these values of MI/PI can be used as predictors for the success rates of the corresponding attacks. Such success rates can be computed for the PI values and simulated for the hypothetical MI ones⁴, as represented in the right part of Figure 4.11. Note that this figure is only illustrative as the connection between the MI/PI and the success rates of attacks using the same models is not proved (in fact, there exist counter-examples [106]). But it was confirmed for many representative case studies⁵, and illustrates (1) how the MI/PI metrics translate into a number of traces to recover the key and (2) how better assumptions generally lead to tighter security guarantees.

Can such bounds be accurate in general? Obtaining bounds on the information loss, such as just described informally, would be highly desirable. But it naturally requires a confirmation that such an approach can be independent of the distributions and statistical tools used for their estimation. Arguing about this issue once again faces the problem that the leakage PDF of cryptographic devices are generally unknown. Yet, one can use a simulated device for this purpose (i.e. a mathematical object for which we generate the leakages according to a known distribution, hence for which we know the MI). For example, we can analyse the MI and PI for a simulated device such that the leakage function deviates from the Hamming weight abstraction, while the evaluator still uses

⁴The techniques in [89, 90] can be used for this purpose. Essentially, the evaluator will use the (well estimated) means of the profiled models with additive Gaussian noise to simulate the traces. He will then adapt the noise level to reach the required MI value and use the corresponding simulated traces to compute success rates.

⁵Including but not limited to [5, 12, 35, 39, 58, 60, 61, 68, 84, 88, 95, 103, 99, 100, 101, 102, 113].

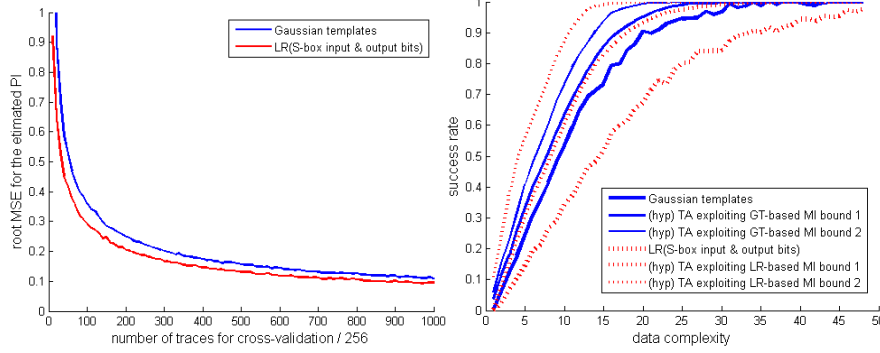


Figure 4.11: Measurements of sample 2605. Left: root MSE for the PI estimates obtained in Figure 4.2. Right: success rates of LR-based and Gaussian template attacks compared with hypothetical template attacks exploiting different MI bounds.

the Hamming weight function as leakage model. As a first step in this direction, we considered the model error as additive and Gaussian, and computed the MI, PI and three MI bounds, as illustrated in the left part of Figure 4.13. We see that for low noise levels, the true leakage model leads to 256 distinguishable events (i.e. $MI = 8$) while the PI is stuck to ≈ 2.5 (i.e. what can be extracted with a Hamming weight model). And as the noise increases, these 256 events first merge into 9 Hamming weight-like events which then become hard to distinguish for standard deviations beyond 10^{-1} . The bounds are given for three noise levels for which the detection of assumption errors is given in the right part of the figure. Two different (negative) observations can be extracted from these experiments. First, many selection rules for the threshold value N_t^{th} could be defined (the figure uses average p -values), and none of the ones we considered provided accurate bounds independent of the noise level. For example, the bound for the standard deviation 10^{-4} does not capture the large MI-PI difference in our simulations. Second, any bound obtained from the MSE will anyway become pessimistic as the physical noise increases (since its impact will eventually dominate the one of assumption errors in the MSE). This is typically observed for the noise level 10^0 in Figure 4.13. As a result, we can conclude that precisely quantifying the information loss due to incorrect assumptions in side-channel attacks is not possible with the techniques presented in this section. Improving this situation is an interesting scope for further (theoretical) research. For example, one could try to establish selection rules

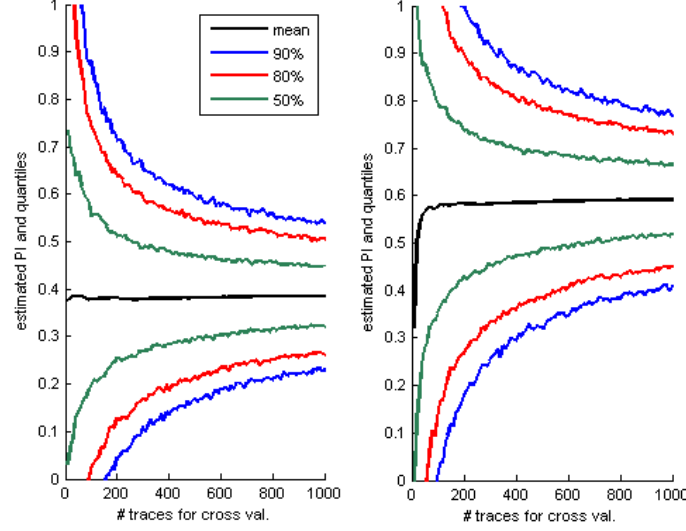


Figure 4.12: Measurements of sample 2605. Quantiles for the PI estimates obtained from the LR-based profiling (left) and Gaussian templates in Figure 4.2.

for N_t^{th} that are well adapted to a class of relevant distributions. One could also exploit blind source separation to better distinguish the contributions of the physical noise and assumption errors in the MSE (but this would require having more observations than sources [59], which may not always be the case, e.g. in power analysis). Yet, in practice these techniques would also drive us away from the goal of analysing cryptographic devices independently of their leakage distributions. In the following, we try to avoid such bottlenecks and conclude this section by suggesting pragmatic guidelines for evaluators.

4.1.6 Pragmatic evaluation guidelines & conclusions

While measuring the information loss due to incorrect assumptions appears difficult in general, the experiments of Figure 4.13 can also be concluded more positively. Namely, they exhibit that as these errors are detected later, the MI-PI difference also decreases. So even if the intuition in the previous section cannot lead to quantitative bounds, it still leads to qualitatively interesting outcomes. In this respect, we believe the tools introduced in this work are essential for the fair evaluation of

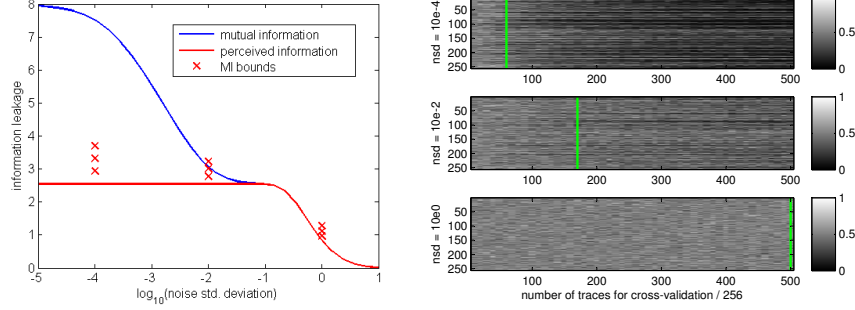


Figure 4.13: Left: mutual information bounds for three noise standard deviations (simulated attacks). Right: detection of assumption errors for the same noise standard deviations.

side-channel attacks, and should be exploited as follows.

First note that the maximum number of measurements in an evaluation is usually determined by practical constraints (i.e. how much time is allowed for the evaluation). Given this limit, estimation and assumption errors can be analysed separately, leading to quantified results such as in Figures 4.2 and 4.6. These steps allow ensuring that the statistical evaluation converged. Next, one should always test the hypothesis that the leakage model is incorrect, as described in Section 4.1.4. Depending on whether assumption errors are detected “early” or “late”, the evaluator should be able to decide whether more refined PDF estimation techniques should be incorporated in his analyses. As discussed in Section 4.1.5, the precise definition of “early” and “late” is hard to formalize in terms of information loss. Yet, later is always better and such a process will at least guarantee that if no such errors are detected *given some measurement capabilities*, an improved model will not lead to significantly improved attacks (since the evaluator will essentially not be able to distinguish the models with this amount of measurements). That is, the proposed methodology can provide an answer to the pragmatic question: “for an amount of measurements performed by a laboratory, is it worth spending time to refine the leakage model exploited in the evaluation?”. In other words, it can be used to guarantee that the security level suggested by a side-channel analysis is close to the worst-case, and this guarantee is indeed conditional to the number of measurements available for this purpose.

Open problems can be envisioned in three main directions. First note

that the most important contribution of this work is to put side-channel evaluations on solid foundations. The separation between estimation and assumption errors is fundamental in this respect. By contrast, the precise procedures we propose for their estimation, as well as the test for detecting incorrect models in Section 4.1.4, are first instances for this purpose. Alternative solutions could certainly be investigated (e.g. to obtain better confidence in the conclusions with less efforts), and constitute an interesting research topic. Besides, the application of the techniques described to more implementations (e.g. protected with countermeasures such as masking), for which the selection of relevant assumptions will be more difficult, is certainly worth further investigations as well. Eventually, the best combination of our tools with dimensionality reduction techniques (or any other preprocessing of the measurements), allowing to efficiently detect/exploit multiple points of interest in leakage traces, is another interesting question.

4.2 Towards easy leakage certification

A practical application of the leakage certification test provided in Section 4.1 requires a number of technical ingredients. Namely, the evaluator first has to characterize the leakages of the target implementation with a sampled (cumulative) distance distribution, and to characterize his model with a simulated (cumulative) distance distribution. Working with distances allows exploiting a univariate goodness-of-fit test even for leakages of large dimensionalities (i.e. it allows comparing the univariate distances between multivariate leakages rather than comparing the multivariate leakages directly). The Cramér-von-Mises divergence is used as a comparison tool in the proposed method. Qualitatively, large divergences between the sampled and simulated distributions essentially mean that the assumptions are imperfect. Quantitatively, the evaluator then has to determine whether such divergences are significant, by verifying whether they can be explained by assumption errors. This essentially requires computing the p -values when testing the hypothesis that the estimated model is correct (which again requires computing many simulated cumulative distance distributions). Summarizing, the beauty of this approach lies in the fact that it only relies on non-parametric estimations and requires no assumptions on the underlying leakage distributions. But this also comes at the cost of quite computationally intensive tools.

In this section, we analyze solutions to mitigate the latter drawback, by investigating whether (computationally) cheaper and (conceptually)

simpler certification procedures can be obtained at the cost of mild assumptions on the statistical distributions in hand. Two natural options directly come to mind for this purpose. They both aim to avoid dealing with the (expensive to characterize) cumulative leakage distributions directly. One possibility is to “summarize” the leakage distribution with its MI/PI estimates (since they can be used as good indicators of the side-channel security level, as now proved in [29]). Another one is to analyse this distribution “moments by moments”, motivated by the recent results in [70]. In both cases, and following the approach in Section 4.1, the main idea remains to compare actual leakage samples generated by a leaking implementation with hypothetical ones generated with the evaluator’s model. Surprisingly, we show that the first approach cannot work, essentially because of situations where model errors in one statistical moment (e.g. the mean) are reflected in another statistical moment (e.g. the variance). This typically arises when using the popular stochastic models (introduced in Section 2.3.2), and actually corresponds to the context of epistemic noise discussed in [46]. More interestingly, we also show that a moment-based approach provides excellent results under reasonable assumptions, and can borrow from the “leakage detection tests” that are already used by evaluation laboratories [38, 66]. The resulting leakage certification method is significantly faster than the one introduced in Section 4.1 (and allows reproducing its experiments). We also show that it easily generalizes to masked implementations, and enables extracting very useful intuitions on the origin of the leakages. Eventually, our new tools additionally lead to simple heuristics to approximate the information loss due to incorrect leakage models, which remained an open problem in the previous method. Summarizing, we simplify leakage certification into a set of easy-to-implement procedures, hopefully more attractive for evaluation laboratories.

4.2.1 A motivating negative result

As mentioned in the introduction of this section, detecting assumption errors is generally more challenging than detecting estimation errors (which is easily done with the previous cross-validation). Intuitively, it requires investigating the likelihood that samples obtained from a leaking device can indeed be explained by an estimated model, which requires a (multivariate) goodness-of-fit test. Since such tests are computationally intensive, an appealing alternative would be to check whether the samples obtained from the leaking device lead to a PI that is at least close enough to the MI: this would guarantee a good estimation of the security level. But, once again, we face the problem that the MI is

unknown, which imposes trying indirect approaches. That is, we would need a metric counterpart to the sampled simulated distance distribution in Section 4.1. This would typically correspond to the following Hypothetical MI (HI):

$$\hat{\text{HI}}(S; X, L) = H[S] + \sum_{s \in \mathcal{S}} \Pr[s] \sum_{x \in \mathcal{X}} \Pr[x] \sum_{l_y^i \in \mathcal{L}_t} \hat{\Pr}_{\text{model}}[l_y^i | s, x] \cdot \log_2 \hat{\Pr}_{\text{model}}[s | x, l_y^i].$$

Intuitively, this HI corresponds to the amount of information that would be extracted from an hypothetical implementation that would exactly leak according to the model $\hat{\Pr}_{\text{model}}$. In itself, the HI is useless to the evaluator, as it is actually disconnected from the chip distribution. For example, even a totally incorrect model (i.e. leading to a negative PI) would lead to a positive HI. By contrast, we could hope that as long as the HI and PI are “close”, the assumption errors are “small enough” for the number of measurements considered in the security evaluation. Furthermore, we could use a simple hypothesis test to detect non-closeness. For a number of traces N in the evaluation set, this would require computing estimates $\hat{\text{PI}}(S; X, L)^{(j)}$ and $\hat{\text{HI}}(S; X, L)^{(j)}$ with cross-validation, and checking whether these estimates come from different (univariate) distributions. If they significantly differ, we would conclude that the model exhibits assumption errors that degrade the estimated security level, in a similar fashion as in Section 4.1.

Unfortunately, and although it can detect certain assumption errors, this approach cannot succeed in general. A simple counter-example can be explained in the context of LR. Say an adversary estimates a model with a linear basis which leads to significant differences between the actual (mean) leakages and the ones suggested by the model. Then, because of the homoscedastic error assumption, the single variance of the LR-based model will reflect this error (i.e. capture both physical noise and model error). As a result, whenever this type of error increases, the PI will decrease (as expected) but the HI will also decrease (contrary to the MI). So testing the consistency between the PI and HI estimates will not reveal the inconsistencies between the PI estimates and the true MI.

4.2.2 A new method to detect assumption errors

Although negative, the previous counter-example suggests two interesting tracks for simplifying leakage certification tests. First, summarizing a complete distribution into representative metrics (e.g. such as the PI)

allows taking advantage of simpler statistical tests. Second, since the fact that the homoscedastic error assumption is not fulfilled implies that errors made in the estimation of certain statistical moments (or more generally, parameters) of a distribution are reflected in other statistical moments of this distribution, a natural approach is to test the relevance of a model “moment by moment”. That is, for a number of traces N in an evaluation set, one could verify that the moments estimated from actual leakage samples are hard to tell apart from the moments estimated from the model (with the same number of samples N). Based on this idea, our simplified method to detect assumption errors will be based on the following two hypotheses (one strictly necessary and the other optional but simplifying).

1. *The leakage distribution is well represented by its statistical moments.* This corresponds to the classical “moment problem” in statistics, for which there exist counter-examples (e.g. the log-normal distribution is not uniquely characterized by its moments). So our (informal) assumption is that these counter-examples will not be significant for our experimental case-studies.
2. *The sampled estimates of our statistical moments are approximately Gaussian-distributed.* This directly derives from the central limit theorem and actually depends on the number of samples used in the estimations (which will become sufficient as the leakages become noisier, e.g. in the case of protected implementations that are most relevant for concrete investigations).

Let us add a couple of motivation words for those assumptions. First recall that we know from the previous results in Section 4.1 that leakage certification is possible without such assumptions, at the cost of somewhat involved statistical reasoning and estimations. So it seems natural to investigate alternative (heuristic) paths allowing us to reach similar conclusions. As will be shown next, this is indeed the case of our simplified approach for a couple of relevant scenarios. Second, statistical moments are at the core of the reasoning regarding the masking counter-measure. That is, the security order of an implementation is generally defined as the lowest informative moment in the leakage distribution (minus one) – see [29] for an extensive discussion of this issue. Besides, many concrete (profiled and non-profiled) side-channel attacks are based (implicitly or explicitly) on parametric PDF estimation techniques that rely on the estimation of moments (e.g. the Gaussian templates and LR-based models in Section 2.3.2, but also second-order attacks such as [24, 83]). So an approach based on an analysis of moments seems well

founded in these cases.⁶ As a result, and maybe most importantly, we believe that the following tools open interesting research avenues regarding the intuitive evaluation of leaking devices based on their moments.

As for the Gaussian assumption, our motivation is even more pragmatic, and relates to the observation that simple t-tests are becoming de facto standards in the preliminary evaluation of leaking devices [38, 66]. So we find it appealing to rely on statistical tools that are already widespread in the CHES community, and to connect them with leakage certification. As will be clear next, this allows us to use the same evaluation method for statistical moments of different orders. However, we insist that it is perfectly feasible to refine our approach by using a well adapted test for each statistical moment (e.g. F-test for variances, ...).

Test specification

The main idea behind of our new leakage certification method is to compare (actual) d th-order moments $\hat{m}_y^d$ estimated from the leakages with (simulated) d th-order moments $\tilde{m}_y^d$ estimated from the evaluator's model $\hat{\text{Pr}}_{\text{model}}$ (by sampling this model). Thanks to our second assumption, this comparison can simply be performed based on Student's t-test. For this purpose, we need multiple estimations of the moments $\hat{m}_y^d$ and $\tilde{m}_y^d$, that we will obtain thanks to an approach inspired from Section 2.3.4 (although there is no cross-validation involved next).

More precisely, we start by splitting the full set of evaluation traces $\mathcal{L}$ into k (non overlapping) sets of approximately the same size $\mathcal{L}^{(j)}$, with $1 \leq j \leq k$. From these k subsets, we produce k estimates of (actual) d^{th} -order moments $\hat{m}_y^{d,(j)}$, each of them from a set $\mathcal{L}^{(j)}$. We then produce a set of simulated traces $\tilde{\mathcal{L}}$ that has the same size and corresponds to the same intermediate values as the real evaluation set $\mathcal{L}$, but where the leakages are sampled according to the model that we want to evaluate. In other words, we first build the model $\hat{\text{Pr}}_{\text{model}} \leftarrow \mathcal{L}$, and then generate a simulated set of traces $\tilde{\mathcal{L}} \leftarrow \hat{\text{Pr}}_{\text{model}}$. Based on $\tilde{\mathcal{L}}$, we produce k estimates of (simulated) d^{th} -order moments $\tilde{m}_y^{d,(j)}$, each of them from a set $\tilde{\mathcal{L}}^{(j)}$, as done for the real set of evaluation traces. From these real and simulated moment estimates, we compute the following

⁶Non-parametric PDF estimations do not suffer from assumption errors (at the cost of a significantly increased estimation cost), so are out of scope here.

quantities:

$$\begin{aligned}\hat{\mu}_y^d &= \hat{\mathbb{E}}_j(\hat{m}_y^{d,(j)}), & \hat{\sigma}_y^d &= \sqrt{\mathbf{v}\hat{\mathbf{a}}\mathbf{r}_j(\hat{m}_y^{d,(j)})}, \\ \tilde{\mu}_y^d &= \hat{\mathbb{E}}_j(\tilde{m}_y^{d,(j)}), & \tilde{\sigma}_y^d &= \sqrt{\mathbf{v}\hat{\mathbf{a}}\mathbf{r}_j(\tilde{m}_y^{d,(j)})},\end{aligned}$$

where $\mathbf{v}\hat{\mathbf{a}}\mathbf{r}$ is the sample variance operator. Eventually, we simply estimate the t statistic (next denoted with Δ_y^d) as follows:

$$\Delta_y^d = \frac{\hat{\mu}_y^d - \tilde{\mu}_y^d}{\sqrt{\frac{(\hat{\sigma}_y^d)^2 + (\tilde{\sigma}_y^d)^2}{k}}}.$$

The p -value of this t statistic within the associated Student's distribution returns the probability that the observed difference is the result of estimation issues, and is computed as:

$$p = 2 \times (1 - \text{CDF}_t(|\Delta_y^d|, \nu)),$$

where CDF_t is the Student's t cumulative distribution function, and ν is its number freedom degrees.⁷ In other words, a small p -value indicates that the model is incorrect with high probability. Concretely, the only parameter to set in this test is the number of non overlapping sets k . Following [29], we used $k = 10$ which is a rather standard value in the literature. Note that increasing k has very limited impact on the accuracy of our conclusions since all variance estimates in the t-test are normalized by k . By contrast it increases the time complexity of the test (so keeping k reasonably small is in general a good strategy).

4.2.3 Simulated experiments

In order to validate our moment-based certification method, we first analyse a couple of simulated experiments where we can control the assumption errors. In particular, and in order to keep these simulations reasonably close to concrete attacks, we consider four distinct scenarios. In the first one (reported in Figure 4.14), both the leakage function Pr_{chip} and the leakage model $\hat{\text{Pr}}_{\text{model}}$ follow a Gaussian distribution, but the model's estimated mean differs from the true distribution. In the second one (reported in Figure 4.15), the leakage function and the leakage model again follow a Gaussian distribution, this time with a model error on the variance. These two examples informally correspond to

⁷Student's t distribution is a parametric probability density function whose only parameter is its number of freedom degrees, that can be directly derived from k and the previous σ estimates as: $\nu = (k - 1) \times [(\hat{\sigma}_y^d)^2 + (\tilde{\sigma}_y^d)^2] / [(\hat{\sigma}_y^d)^4 + (\tilde{\sigma}_y^d)^4]$.

the context of LR-based attacks where the basis is not large enough to capture the exact mean values. (Yet, as previously discussed and will be clear from the measured experiments in the next section, both errors usually happen jointly in those cases). Our third and fourth examples correspond to a slightly different setting aimed to emulate a masked implementation for which the true distribution is typically a mixture [102]. So, in the third scenario (reported in Figure 4.16), the leakage function has a Gaussian mixture distribution with a non-zero skewness, while the leakage model is still a Gaussian approximation. And in the fourth scenario (reported in Figure 4.17), the leakage function has a Gaussian mixture distribution with a non-zero kurtosis, while the leakage model is still a Gaussian approximation. In any case, we represent the true distribution and the biased model, the estimated moments for these two distributions, and the p -value of our certification test.

The results are mostly as expected and confirm the simplicity of the method. That is, as the number of measurements in the evaluation set increases, we are able to detect the assumption errors in all the cases. The only difference between the applications to different moments is that errors on higher-order moments may be more difficult to detect as the noise increases. This difference is caused by the same argument that justifies the relevance of the higher-order masking countermeasure. Namely, the sampling complexity when estimating the moments of a distribution increases exponentially in d . However, this is not a limitation of the certification test: if such errors are not detected for a given evaluation set, it just means that their impact is still small in front of assumption errors at this stage of the evaluation. Besides, we note that the respective relevance of the model errors on different moments will be further discussed in Section 4.2.5.

4.2.4 Measured experiments

In order to obtain a fair comparison with the results provided in Section 4.1, we also applied our new leakage certification method to the same case-study. That is, we used the measurement setup from Section 2.3.1 and evaluated the relevance of two important profiling methods, namely the Gaussian TA and LR, for the most informative time sample in our leakage traces (i.e. the sample 2605, with maximum PI).

The main difference with the previous simulated experiments is that we now have to test 256 models independently (each of them corresponding to a target intermediate value $y = x \oplus s$). Our results are represented in Figure 4.18 where we plot the p -values output by our different t-tests

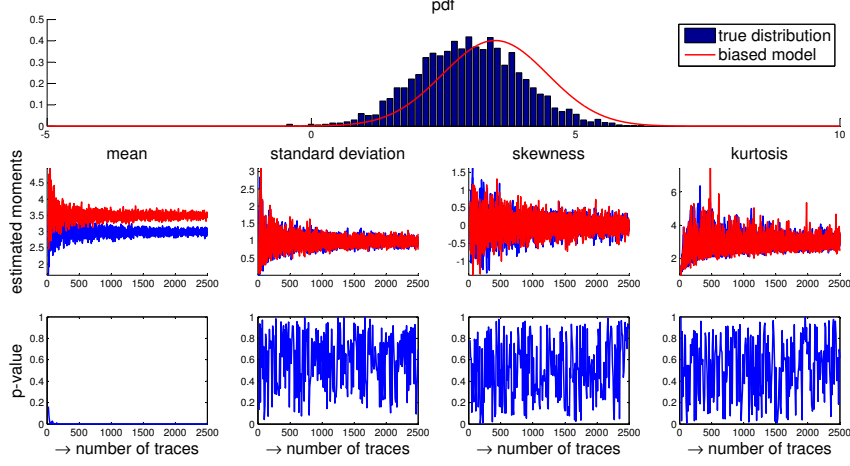


Figure 4.14: Gaussian leakages, Gaussian model, error in the estimated mean.

for four statistical moments (i.e. the mean, variance, skewness and kurtosis). A look at the first two moments essentially confirms the results of the previous section. More precisely, the Gaussian templates seem to capture the measured leakages quite accurately (for the 256,000 traces in our evaluation set). By contrast, the linear regression quickly exhibits inconsistencies. Interestingly, assumption errors appear both in the means and in the variances, which corresponds to the expected intuition. That is, errors in the means are detected because for most target intermediate values, the actual leakage cannot be accurately predicted by a linear combination of the S-box output bits. And errors in the variances appear because the LR-based models rely on the homoscedastic error assumption and capture both physical noise and noise due to assumption errors in a single term.

By contrast, and quite intriguingly, a look at the last two moments (i.e. skewness and kurtosis) also shows some differences with the results in Section 4.1. That is, we remark that even for Gaussian templates, small model errors appear in these higher-order moments. This essentially corresponds to the fact that our measured leakages do not have perfectly key-independent skewness and kurtosis, as we assume in Gaussian PDF estimations. This last observation naturally raises the question whether these errors are significant, i.e. do they contradict the results of

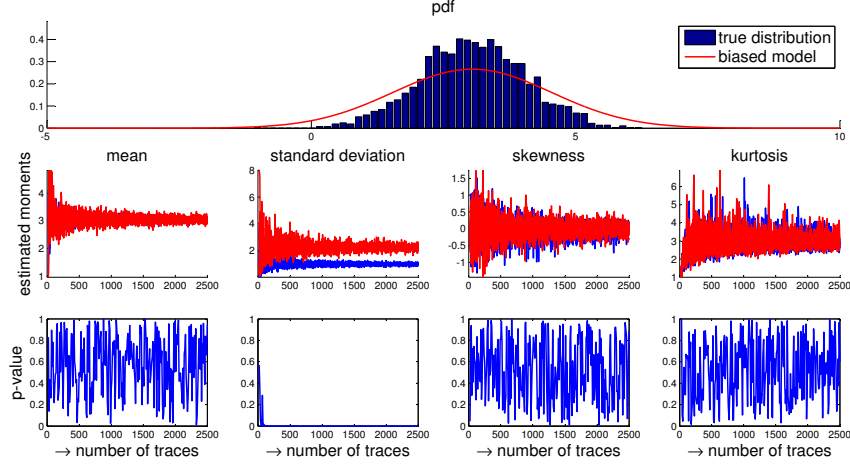


Figure 4.15: Gaussian leakages, Gaussian model, error in the estimated variance.

the previous leakage certification test? In the next section, we show that it is not the case, and re-conciliate both approaches by investigating the respective informativeness of the four moments in our new test.

4.2.5 Quantifying the information loss

Since Figure 4.18 suggests the existence of (small) model errors in our Gaussian templates that are due to an incorrect characterization of the third- and fourth-order moments in our leakage traces, we now want to investigate whether these errors are leading to significant information losses. Fortunately, our “per-moment” approach to leakage certification also allows simple investigations in this direction (which incidentally and heuristically answers one of the open questions in Section 4.1, about the information loss due to model errors). In particular, we can simply use the MCP-DPA mentioned in Section 2.3.3 for this purpose. Roughly, this tool computes the correlation between a simplified model (that corresponds to d^{th} -order moments of the leakage distribution) to samples raised to the power d (possibly centered or standardized if we consider centered and standardized moments). As discussed in [70], the resulting estimated correlation coefficient features a “metric intuition”: the higher the value of the MCP-DPA distinguisher computed for an order d , the more efficient the MCP-DPA attack exploiting this statistical or-

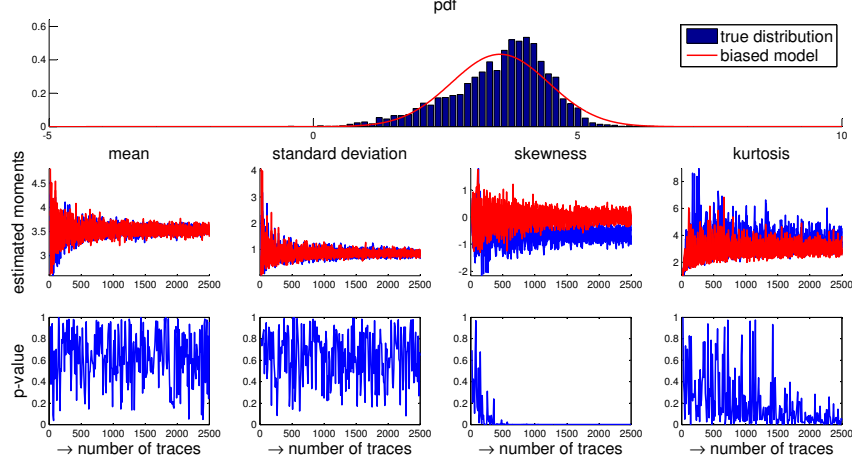


Figure 4.16: Gaussian mixture leakages, Gaussian model, error in the estimated skewness.

der of the actual leakage distribution. Hence, computing the value of the MCP-DPA distinguisher for different values of d should solve our problem.

Concretely, we start by applying MCP-DPA in the traditional sense and exploit cross-validation for this purpose, this time following exactly Section 2.3.4. That is, the set of evaluation traces $\mathcal{L}$ is again split into k (non overlapping) sets $\mathcal{L}^{(i)}$ of approximately the same size, and we use profiling sets $\mathcal{L}_p^{(j)} = \bigcup_{i \neq j} \mathcal{L}^{(i)}$ and test sets $\mathcal{L}_t^{(j)} = \mathcal{L} \setminus \mathcal{L}_p^{(j)}$. We then repeatedly compute the d^{th} -order moments $\hat{m}_y^{d,(j)} \leftarrow \mathcal{L}_p^{(j)}$, and the d^{th} -order MCP-DPA distinguisher:

$$\text{MCP-DPA}^{(j)}(d) = \hat{\rho}\left(\hat{M}_Y^{d,(j)}, (L_y)^d \leftarrow \mathcal{L}_t^{(j)}\right).$$

As previously mentioned, it corresponds to the sample correlation between the random variable representing the estimated moments $\hat{M}_Y^d$, and the random variable corresponding to the leakage samples coming from the test set $L_y \leftarrow \mathcal{L}_t^{(j)}$, raised to power d (possibly centered or standardized if we consider centered and standardized moments). The $k = 10$ estimates for this MCP-DPA metric are represented in the top part of Figure 4.19. We additionally considered two slightly tweaked versions of MCP-DPA, where we rather estimate Gaussian TA (resp.

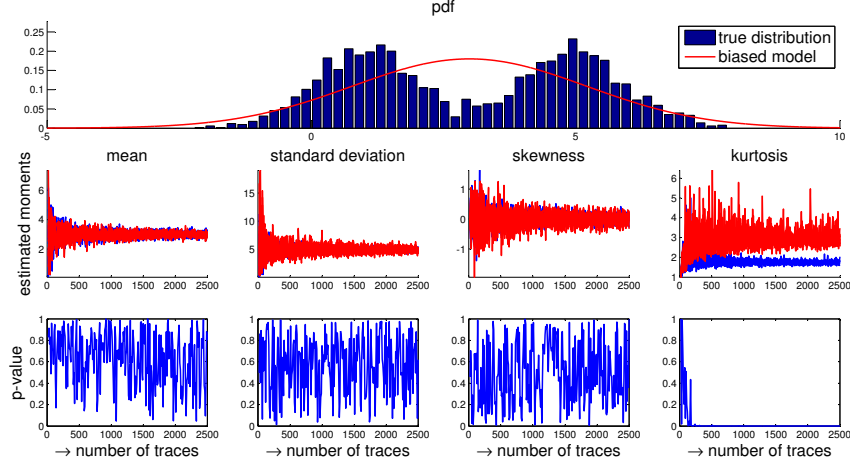


Figure 4.17: Gaussian mixture leakages, Gaussian model, error in the estimated kurtosis.

LR-based) models $\hat{P}_{\text{model}}^{\text{ta}}$ (resp. $\hat{P}_{\text{model}}^{\text{lr}}$), and consider the two (resp. one) key-dependent moments from these models to compute the metric. These tweaked MCP-DPAs are represented in the middle (resp. lower) part of the figure.

Our main observations are as follows. First, the upper part of the figure suggests that the most informative moments in our leakage traces are the mean and variance. There is indeed a small amount of information in the skewness and kurtosis. But by considering the classical rule-of-thumb that the measurement complexity of a correlation-based attack is inversely proportional to the square of its correlation coefficient, we can see that the additional information gain in these higher-order moments is very limited in our context. This observation backs up the conclusions of the generic leakage certification test in Section 4.1 that Gaussian templates are sufficiently accurate for our test set, even though the noise is apparently not completely Gaussian (which is a usual assumption in the literature). It can be similarly quantified by using the following approximation from [65]:

$$I(X, Y) \approx -\frac{1}{2} \times \log_2 \left(1 - \rho(X, Y)^2 \right),$$

considering that attacks exploiting two moments would take advantage of the sum of their respective information (i.e. considering them as in-

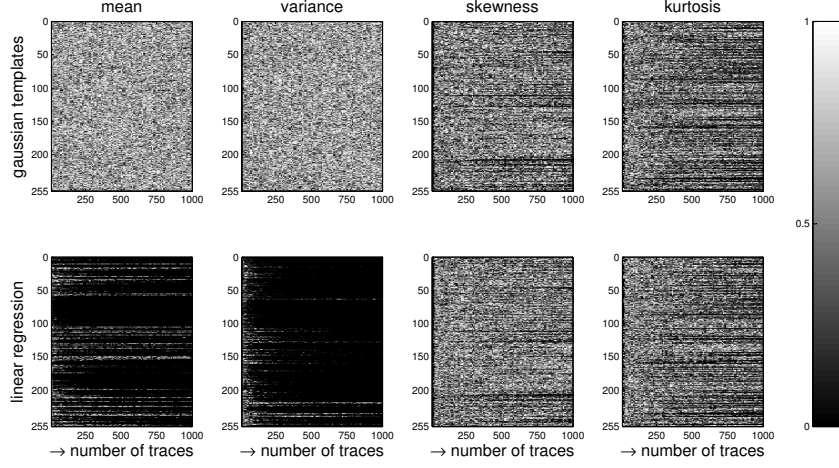


Figure 4.18: Results of the new leakage certification test for actual measurements.

dependent information channels), and using the formula in [29] which shows that the measurement complexity of a side-channel attack is inversely proportional to the MI/PI leaked by a target implementation. Next, we also see that TA-based and LR-based MCP-DPA yield no information in the higher-order moments, which trivially derives from the fact that they rely on a Gaussian assumption. Eventually, we notice that the information loss between LR-based models and TA-based models can be approximated thanks to the correlation between their moments. For example, and considering the means in Figure 4.19, we can compute the value of the LR-based MCP-DPA distinguisher – worth ≈ 0.48 in the figure – by multiplying the value of the TA-based MCP-DPA distinguisher – worth ≈ 0.74 in the figure – by $\hat{\rho}(\hat{M}_Y^{d,ta}, \hat{M}_Y^{d,lr})$ – worth ≈ 0.65 in our experiments (i.e. by taking advantage of the “product rule” for the correlation coefficient in [107]).

Those final tools are admittedly informal. Yet, we believe they provide a useful variety of heuristics allowing evaluators to analyse the results of their certification tests. In particular, they lead to easy-to-exploit intuitions regarding the impact (or lack thereof) of model errors detected in moments of a given order. As discussed in the beginning of Section 4.2.2, further formalizing these findings, and possibly putting forward relevant scenarios where our simplified approach leads to signif-

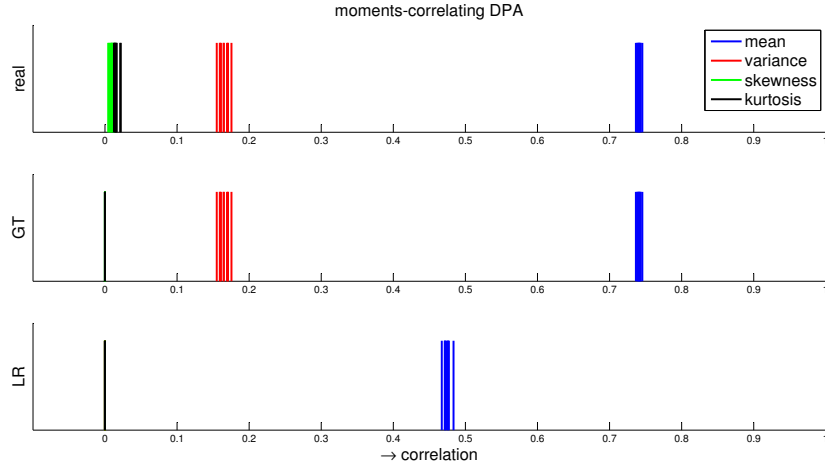


Figure 4.19: MCP-DPA results actual measurements.

icant shortcomings, is an interesting scope for further research.

Chapter 5

Other work

Since I started my PhD thesis, I have had the opportunity to work with different researchers on various research topics. This led to the publication of papers that were the product of these collaborations, and that were approved by the research community in multiple international conferences. Doing this PhD also allowed me to go to San Francisco for a 3-month internship at Cryptography Research Inc. This chapter briefly lists the publications that are sorted by theme, namely hardware implementations, intellectual property protection and side-channel attacks. Their abstract is given as a short summary, followed by a brief description of my personal contribution. A short summary of my experience at CRI is eventually given.

5.1 Hardware implementations

Compact FPGA Implementations of the Five SHA-3 Finalists

Authors: Stéphanie Kerckhof, François Durvaux, Nicolas Veyrat-Charvillon, Francesco Regazzoni, Gueric Meurice de Dormale and François-Xavier Standaert

Published in *Smart Card Research and Advanced Applications - 10th International Conference, CARDIS 2011*, Leuven, Belgium, September 14-16, 2011.

Abstract. Allowing good performances on different platforms is an important criteria for the selection of the future SHA-3 standard. In this paper, we consider the compact implementations of BLAKE, Grøstl, JH, Keccak and Skein on recent FPGA devices. Our results bring an interesting complement to existing analyses, as the previous work on

FPGA implementations of the SHA-3 candidates was for the most optimized for high throughput applications. Following recent guidelines for the fair comparison of hardware architectures, we put forward clear trends for the selection of the future standard. First, compact FPGA implementations of Keccak are less efficient than their high throughput counterparts. Second, Grøstl shows interesting performances in this setting, in particular in terms of throughput over area ratio. Third, the remaining candidates are comparably suitable for compact FPGA implementations, with some slight contrasts (in area cost and throughput).

Personal contribution. For this work, I was in charge of implementing the BLAKE hash function algorithm. It allowed me to acquire a good practice for implementing cryptographic algorithms with reasonably realistic constraints, and to understand how they work and the implementation trade-off they offer. I was also in charge of presenting our implementation results at the CARDIS 2011 conference.

Towards Green Cryptography: A Comparison of Lightweight Ciphers from the Energy Viewpoint

Authors: Stéphanie Kerckhof, François Durvaux, Cédric Hocquet, David Bol and François-Xavier Standaert

Published in *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop*, Leuven, Belgium, September 9-12, 2012.

Abstract. We provide a comprehensive evaluation of several lightweight block ciphers with respect to various hardware performance metrics, with a focus on the energy cost. This case study serves as a background for discussing general issues related to the relative nature of hardware implementation comparisons. We also use it to extract intuitive observations for new algorithm designs. Implementation results show that significant differences can be observed between lightweight ciphers, in particular when considering both encryption and decryption architectures, and the impact of key scheduling algorithms. Yet, these differences are moderated when looking at their amplitude, and comparing them with the impact of physical parameters tuning, e.g. frequency / voltage scaling.

Personal contribution. For this second paper on cryptographic algorithms implementations, I was in charge of implementing the ICEBERG and KATAN block cipher algorithms. In addition to the results provided in this paper, these implementations were eventually realized as

an ASIC. I took part in the fabrication process, which allowed me to get some practice about the development of integrated circuits.

SleepWalker: A 25-MHz 0.4-V Sub-mm² 7- μ W/MHz Microcontroller in 65-nm LP/GP CMOS for Low-Carbon Wireless Sensor Nodes

Authors: David Bol, Julien De Vos, Cédric Hocquet, François Botman, François Durvaux, Sarah Boyd, Denis Flandre and Jean-Didier Legat

Published in *IEEE Journal of Solid-State Circuits (JSSC)*, volume 48, pp. 20-32, 2013.

Abstract. Integrated circuits for wireless sensor nodes (WSNs) targeting the Internet-of-Things (IoT) paradigm require ultralow-power consumption for energy-harvesting operation and low die area for low-cost nodes. As the IoT calls for the deployment of trillions of WSNs, minimizing the carbon footprint for WSN chip manufacturing further emerges as a third target in a design-for-the-environment (DfE) perspective. The SleepWalker microcontroller is a 65-nm ultralow-voltage SoC based on the MSP430 architecture capable of delivering increased speed performances at 25 MHz for only 7 μ W/MHz at 0.4 V. Its sub-mm² die area with low external component requirement ensures a low carbon footprint for chip manufacturing. SleepWalker incorporates an on-chip adaptive voltage scaling (AVS) system with DC/DC converter, clock generator, memories, sensor and communication interfaces, making it suited for WSN applications. An LP/GP process mix is fully exploited for minimizing the energy per cycle, with power gating to keep stand-by power at 1.7 μ W. By incorporating a glitch-masking instruction cache, system power can be reduced by up to 52%. The AVS system ensures proper 25-MHz operation over process and temperature variations from -40°C to $+85^{\circ}\text{C}$, with a peak efficiency of the DC/DC converter above 80%. Finally, a multi- V_t clock tree reduces variability-induced clock skew by $3 \times$ to ensure robust timing closure down to 0.3 V.

Personal contribution. During this project, I had to develop some of the microcontroller peripherals, such as the timers or the time-to-digital converter. The former are used, for instance, for controlling the cycles while the microcontroller sleeps in order to save its energy. The latter is the interface with the outside world, namely the sensors, since this project is intended for sensors network applications.

5.2 Intellectual property protection

Intellectual Property Protection for Integrated Systems Using Soft Physical Hash Functions

Authors: François Durvaux, Benoît Gérard, Stéphanie Kerckhof, François Koeune and François-Xavier Standaert

Published in *Information Security Applications - 13th Intl. Workshop, WISA 2012*, Jeju Island, Korea, August 16-18, 2012.

Abstract. Intellectual property right violations are an important problem for integrated system designers. We propose a new solution for mitigating such violations, denoted as soft physical hash functions. It combines previously introduced ideas of soft hash functions (in the field of image processing) and side-channel leakage (in the field of cryptographic hardware). For this purpose, we first introduce and formalize the components of an intellectual property detection infrastructure using soft physical hash functions. Next, we discuss its advantages over previous proposals aiming at similar goals. The most important point here is that the proposed technique can be applied to already deployed products. Finally, we validate our approach with a first experimental study.

Personal contribution. Because it was the first work to propose this specific solution (non-invasive and passive side-channel analysis) in order to identify intellectual properties, all the steps were made collaboratively with the other authors. I took part in the framework specification, the instantiation we used for this 8-bit case-study, and the experimentation process. I was in charge of presenting our results at the WISA 2012 conference.

Intellectual property protection for FPGA designs with soft physical hash functions: First experimental results

Authors: Stéphanie Kerckhof, François Durvaux, François-Xavier Standaert and Benoît Gérard

Published in *IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013*, Austin, TX, USA, June 2-3, 2013.

Abstract. The use of Soft Physical Hash (SPH) functions has recently been introduced as a flexible and efficient way to detect Intellectual Property (IP) cores in microelectronic systems. Previous work mainly investigated software IP to validate this approach. In this paper, we extend it towards the practically important case of FPGA designs. Based

on experiments, we put forward that SPH function-based detection is a promising and low-cost solution for preventing anti-counterfeiting, as it does not require any a-priori modification of the design flow. In particular, we illustrate its performances with stand-alone FPGA designs, re-synthesized FPGA designs, and in the context of parasitic IPs running in parallel.

Personal contribution. This work is a natural extension of the WISA 2012 paper to hardware implementations of block cipher algorithms. Hence, I took part in the instantiation of the detection framework as well as in the experimentation procedure.

Support Vector Machines for Improved IP Detection with Soft Physical Hash Functions

Authors: Ludovic-Henri Gustin, François Durvaux, Stéphanie Kerckhof, François-Xavier Standaert and Michel Verleysen

Published in *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014*, Paris, France, April 13-15, 2014.

Abstract. Side-channel analysis is a powerful tool to extract secret information from microelectronic devices. Its most frequently considered application is destructive, i.e. key recovery attacks against cryptographic implementations. More recently, it has also been considered constructively, in the context of intellectual property protection/detection, e.g. through the use of side-channel based watermarks or soft physical hash functions. The latter solution is interesting from the application point-of-view, because it does not require any modification of the designs to protect (hence it implies no performance losses). Previous work in this direction exploited simple (correlation-based) statistical tools in different (more or less challenging) scenarios. In this paper, we investigate the use of support vector machines for this purpose. We first argue that their single-class extension is naturally suited to the problem of intellectual property detection. We then experimentally show that they allow dealing with more complex scenarios than previously published, hence extending the relevance and applicability of soft physical hash functions.

Personal contribution. This last paper about intellectual property protection reports the results obtained by a master student during his master thesis. I supervised his work along with other researchers. We were also in charge of building his measurement sets with the appropriate parameters. It allowed me to understand machine learning techniques

such as the support vector machine, and how they can be applied in the context of side-channel analysis.

5.3 Side-channel attacks

Efficient Removal of Random Delays from Embedded Software Implementations Using Hidden Markov Models

Authors: François Durvaux, Mathieu Renaud, François-Xavier Standaert, Loïc van Oldeneel tot Oldenzeel and Nicolas Veyrat-Charvillon

Published in *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012*, Graz, Austria, November 28-30, 2012.

Abstract. Inserting random delays in cryptographic implementations is often used as a countermeasure against side-channel attacks. Most of the previous work on the topic focuses on improving the statistical distribution of these delays. For example, efficient random delay generation algorithms were proposed at CHES 2009/2010. These solutions increase security against attacks that solve the lack of synchronization between different leakage traces by integrating them. In this paper, we demonstrate that integration may not be the best tool to evaluate random delay insertions. For this purpose, we first describe different attacks exploiting pattern-recognition techniques and Hidden Markov Models. Using these tools and as a case study, we perform successful key recoveries against an implementation of the CHES 2009/2010 proposal in an Atmel microcontroller, with the same data complexity as against an unprotected implementation of the AES Rijndael. In other words, we completely cancel the countermeasure in this case. Next, we show that our cryptanalysis tools are remarkably robust to attack improved variants of the countermeasure, e.g. with additional noise or irregular dummy operations. We also exhibit that the attacks remain applicable in a non-profiled adversarial scenario. These results suggest that the use of random delays may not be effective for protecting small embedded devices against side-channel leakage. They highlight the strength of Viterbi decoding against such time-randomization countermeasures, in particular when combined with a precise description of the target implementations, using large lattices.

Personal contribution. This work was the opportunity for me to become familiar with signal processing technique in the context of side-channel attacks. I was in charge of the preliminary experiments for

removing the random delay countermeasure, which resulted in a successful attack. I took also part in the reasoning in order to design the advanced solution based on hidden markov models that is used in the following of that paper and provides better results. Finally, I was in charge of presenting our results at the CARDIS 2012 conference.

5.4 Internship at Cryptography Research Inc.

When I arrived at CRI for my 3-month internship, I was directly integrated into the DPA team under the supervision of Dr. Pankaj Rohatgi and Dr. Mike Tunstall. During this period, I contributed to the development of a new masking scheme. Concretely, I worked on a proof-of-concept demonstrating (i) that multiplicative masking can be made resistant to the zero-value DPA (i.e. that exploits the particular case of a zero-absorbing value), and (ii) that first-order DPA resistant implementations of first-order masking schemes can also be written in the C programming language. It resulted in my nomination as co-inventor of the patent that was filed for this new scheme (no reference is yet publicly available).

Chapter 6

Conclusion and perspectives

Providing fair side-channel security evaluations raises many questions and challenges. In this thesis, we aimed to provide answers and solutions in this direction by investigating different aspects in the side-channel analysis flow. Concretely, in a first contribution, we explored leakage detection and points-of-interest detection procedures, and showed how they significantly affect the evaluation/attack outcome. We also demonstrated that these two procedures are in fact connected. As a second contribution, we developed leakage certification methods that allow an evaluator to evaluate the models that he built during the modelling step: one that formalizes the problem and provides a sound statistical analysis with a non-parametric approach, and the other that simplifies the method in order to provide better intuitions, but at the cost of a couple of heuristic assumptions. Both parts led to various and interesting results that are briefly summarized in this last chapter. We then discuss the future work and perspectives open by our results.

On the leakage and points-of-interest detection

Detecting the leakages corresponds to determining if there is any leaking information about the processed data, regardless of its exploitability. Detecting the points-of-interest is a complementary task that is used in order to identify the most informative time samples, i.e. the points that can be exploited by the adversary to lead his attack. In Chapter 3, we explored and analysed these steps and showed that even though they are different tasks with (at first) different goals, they are in fact connected. In this regard, our contribution is threefold.

We led our first investigations on the widely spread CRI's non-specific “fixed vs. random” t-test used for leakage detection. We aimed

to provide a better understanding of this test and the underlying leakage detection process. For this purpose, we introduced an alternative test denoted as the ρ -test, which essentially features the Pearson’s correlation coefficient and the Fisher transformation. The correlation was selected for its simplicity and because intuitions can easily be extracted. Moreover, working with the Fisher transformation allowed us to directly compare the detection results output by the two methods. We established our first observations with a simulated setup and further validated them with real measurements. The main message about this part of the reported work is that, at the end, the sampling complexity of the test outcome does not really depend on the choice of the statistic but rather on the signal-to-noise ratio that can be computed on the leakage traces. Yet, the ρ -test, as we instantiated, suffers from the estimation of the model. Besides, the “fixed vs. random” t-test works better, only if it is applied on the whole AES (whole plaintext and whole execution), which relies on the fact that extreme values (i.e. the ones maximizing the SNR) will be manipulated at some point (by chance, since the key remains secret). Moreover, while the t-test only answers whether leakages are present or not (regardless of their exploitability), the ρ -test adds up the available information – which relates to the POI detection process – and can be directly extended to higher statistical moments. Furthermore, we showed that our reasoning based on the SNR not only allows a better statistical understanding of leakage detection, but can also lead to more efficient t-tests. Namely, it directly suggests that if the evaluator’s goal is to minimize the number of samples needed to detect data-dependent information in side-channel measurements, considering a partitioning based on two fixed plaintexts (rather than one fixed and one random plaintext) leads to significantly faster detection speeds.

As a second contribution on this topic, we explored the use of the projection pursuit algorithms in order to detect and project the points-of-interest in leakage traces. The methods provided in the literature (e.g. PCA and LDA) do not allow working with countermeasures such as the masking that splits the information onto different time samples (within a reasonable amount of data). In general, projection pursuits are heuristic algorithms that track the improvement (or the lack thereof) of an objective function by applying small random modifications on the best projection found so far. For this part of the reported work, we considered two case-studies: (i) an unprotected implementation of the AES in order to validate the approach, and (ii) a (precomputed) masked version of the AES S-box:

1. Regarding the unprotected setting, we made use of a linear pro-

jection of the time samples. The weights (associated to every time sample) are modified such that the objective function is increased. We considered two objective functions: the SNR and a profiled CPA. The former was used in order to have comparable results with the state-of-the-art (specifically with the LDA). The latter was used to validate the metric for a further extension to the masking case. Experimental results showed that both objective functions equally provide a significant improvement of a template attack when attacking the projected samples, in comparison to the same attack on the best available (non-projected) time sample. This confirmed our claim that in this context, the projection pursuits provide as comparable projections as the LDA dimensionality reduction technique, although heuristic.

2. Regarding the masked implementation setup, the challenge was to find a group of points that contained information on all the shares. For this purpose, we slightly modified the projection pursuit instantiation proposed for the unprotected setting, and made use of a window-based local search algorithm. Regions-of-interest (i.e. where the POIs lie) are found by moving and then refining the windows. We used an MMPC as objective function. This statistic targets the higher statistical moments, i.e. the moments that carry the information after combining the time samples. With our results, we showed that this method allows gaining a constant – but practically significant – factor over the previous methods proposed in the literature in order to find the POIs in this setting. Moreover, the algorithm can be extended to any order of masking, simply by considering as many windows as the number of shares. Yet, the proposed algorithms are heuristic: they need to set a lot of parameters, for which we provided guidelines. By contrast with the unprotected setting, this extended search algorithm allows the evaluator to find the POIs corresponding the different shares, but does not combine them in order to maximize the extracted information.

Finally, in the last part of our work on this topic, we discussed the use of the ρ -test as an objective function in our POI detection automated tool. The experiments featured the same masked setup and the same algorithm instantiation. Yet, in this contribution, we showed how the previous methods and tools can be extended to an unprofiled setting at the cost of also finding some useless points dependent on the plaintext only. The purpose of this contribution is mainly to raise awareness that up to the selection of POIs in leakage traces, side-channel security

evaluations can essentially be performed in a black box way, and without any key profiling.

To summarize, we may conclude that leakage and points-of-interest detection methods are important topics of research. An improper use may significantly affect the evaluation/attack outcome. It is very important for the evaluator to understand the statistics behind the tools he is using. Otherwise, he may underestimate the amount of information that is possibly accessible to the adversary. Regarding our contribution, a future scope of research would be to investigate a combination of the masked implementation POIs. Indeed, in the method proposed so far, only the POIs are found, but no combination in order to maximize the extracted information is provided. This work would require investigating different projection and objective functions. It may also be interesting to explore other strategies to roam the leakage traces in order to find the POIs. The proposed method is heuristic, hence depends on lots of parameters that may be changed in order to possibly enhance its functioning. Finally, it may be worth investigating other masking schemes that have different structures (i.e. that leak differently) and that may not be based on precomputed tables.

On the leakage certification

The goal of an evaluator is to accurately determine the worst-case side-channel security of an implementation, i.e. the amount of information that is leaked by the device. For this purpose, he ideally needs a leakage model that exactly corresponds to the true leakage distribution, with a Bayesian distinguisher. Yet, in practice, such perfect models are generally unknown, and density estimation techniques have to be used to approximate the leakage distribution. This raises the fundamental problem that all security evaluations are potentially biased by both estimation and assumption errors. In Chapter 4, we provided and implemented methodological tools in order to solve this issue.

In a first part of the reported work on this topic, we tackled both estimation and assumption errors independently. Regarding the former, we proposed to make use of the cross-validation technique that allows generating multiple independent results of perceived information. Their spread indicates whether the statistic is properly estimated, or not. In order to answer the assumption issue, we proposed a method taking advantage of a hypothesis test for which the null hypothesis states that the model fits the observations. This test features two ingredients: (i) a distance sampling technique that allows us to capture all the char-

acteristics of a distribution (even high-order ones) by computing the distances between pairs of samples, and (ii) a non-parametric goodness-of-fit test that allows us to determine whether the null hypothesis is verified through p -values. We validated our approach by comparing Gaussian templates and linear regression models on measurements of the AES running on an 8-bit microcontroller. With the proposed method, it is possible to detect both kinds of errors and determine if the evaluator’s model is affected by one or the other. This first part of our contribution on this topic allows us to formalize the problem. It also gives a first sound and non-parametric solution to a very practical problem that may be faced in the security evaluation procedures. Yet, to the question “How good is my leakage model?”, it gives a qualitative answer, and not a quantitative one.

In the following, we extended the previous leakage certification procedure. We showed that the assumption error detection can be significantly simplified at the cost of mild assumptions: (i) the distributions are well characterized by their moments, and (ii) the moment estimations are Gaussian distributed. This allows the evaluator to perform a moment-based comparison between the model and the true distribution. The Student’s t -test is used for this purpose, and the related p -value is extracted for every difference between a pair of moments. Similarly to the previous method, the p -value gives the probability for the model to correspond to the actual observations. Since the moments are treated independently, this method also helps the evaluator to determine which ones are failing. Therefore, he has the possibility to refine his model. However, not all the moments contain the same amount of information. We then proposed a further investigation exploiting MMPC that allows the evaluator to quantify this information in every moments, and to focus on the most relevant ones. Overall, this new method quite significantly simplifies our first proposal. It also gives more insight to the evaluator in his selection of an appropriate leakage model. If the model fails, the amount of information in every moment determines whether it is worth improving the model, or not. This method also allows the evaluator to set sound bounds on the total information that may be accessed by the adversary.

To conclude, this work is the first to provide a solution to distinguish and treat the estimation and assumption errors that are likely to be encountered during a security evaluation. The most important contribution is that we put side-channel evaluations on solid statistical foundations. The separation between estimation and assumption errors is fundamental in this respect. Regarding the possible future research

directions, the application of the described techniques to more implementations (e.g. protected with countermeasures such as masking), for which the selection of relevant assumptions will be more difficult, is definitely worth the while. The combination of our tools with dimensionality reduction techniques (or any other preprocessing of the measurements), allowing to efficiently detect/exploit multiple points of interest in leakage traces, is another interesting question.

References

- [1] Theodore W. Anderson. On the distribution of the two-sample Cramer-von Mises criterion. *The Annals of Mathematical Statistics*, 33(3):1148–1159, 1962.
- [2] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In Goubin and Matsui [40], pages 1–14.
- [3] Atmel. AVR microcontrollers - ATmega644P. <http://www.atmel.com/devices/atmega644p.aspx>.
- [4] Josep Balasch, Baris Ege, Thomas Eisenbarth, Benoît Gérard, Zheng Gong, Tim Güneysu, Stefan Heyse, Stéphanie Kerckhof, François Koeune, Thomas Plos, Thomas Pöppelmann, Francesco Regazzoni, François-Xavier Standaert, Gilles Van Assche, Ronny Van Keer, Loïc van Oldeneel tot Oldenzeel, and Ingo von Maurich. Compact implementation and performance evaluation of hash functions in ATtiny devices. In Mangard [63], pages 158–172.
- [5] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, and Ingrid Verbauwhede. Theory and practice of a leakage resilient masking scheme. In Wang and Sako [115], pages 758–775.
- [6] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. To appear in the proceedings of CARDIS 2015.
- [7] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer’s apprentice guide to fault attacks. Cryptology ePrint Archive, Report 2004/100, 2004. <http://eprint.iacr.org/>.

- [8] Paulo S.L.M. Barreto and Vincent Rijmen. The KHAZAD legacy-level block cipher. Primitive submitted to Nessie, available at <http://www.larc.usp.br/~pbarreto/KhazadPage.html>, 2000.
- [9] Robert Bartoszynski, Dennis K. Pearl, and John Lawrence. A multidimensional goodness-of-fit test based on interpoint distances. *Journal of the American Statistical Association*, 92(438):pp. 577–586, 1997.
- [10] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *J. Cryptology*, 24(2):269–291, 2011.
- [11] Lejla Batina, Jip Hogenboom, and Jasper G. J. van Woudenberg. Getting more from PCA: First results of using principal component analysis for extensive power analysis. In Dunkelman [30], pages 383–397.
- [12] Ali Galip Bayrak, Francesco Regazzoni, Philip Brisk, François-Xavier Standaert, and Paolo Ienne. A first step towards automatic application of power analysis countermeasures. In Leon Stok, Nikil D. Dutt, and Soha Hassoun, editors, *DAC*, pages 230–235. ACM, 2011.
- [13] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
- [14] Indra M. Chakravarti, Radha G. Laha, and J. Roy. *Handbook of methods of applied statistics*, Volume I, John Wiley and Sons, pp. 392–394, 1967.
- [15] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Wiener [118], pages 398–412.
- [16] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA*,

- USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
- [17] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013.
- [18] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *LNCS*, pages 252–263. Springer, 2000.
- [19] Christophe Clavier, Jean-Luc Danger, Guillaume Duc, M.Abdelaziz Elaabid, Benoît Gérard, Sylvain Guilley, Annelie Heuser, Michael Kasper, Yang Li, Victor Lomné, Daisuke Nakatsu, Kazuo Ohta, Kazuo Sakiyama, Laurent Sauvage, Werner Schindler, Marc Stöttinger, Nicolas Veyrat-Charvillon, Matthieu Walle, and Antoine Wurcker. Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest. *Journal of Cryptographic Engineering*, pages 1–16, 2014.
- [20] Christophe Clavier and Kris Gaj, editors. *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*. Springer, 2009.
- [21] Jeremy Cooper, Elke De Mulder, Gilbert Goodwill, Josh Jaffe, Gary Kenworthy, and Pankaj Rohatgi. Test Vector Leakage Assessment (TVLA) methodology in practice (extended abstract). ICMC 2013. <http://icmc-2013.org/wp/wp-content/uploads/2013/09/goodwillkenworthtestvector.pdf>.
- [22] Jean-Sébastien Coron and Ilya Kizhvatov. An efficient method for random delay generation in embedded software. In Clavier and Gaj [20], pages 156–170.
- [23] Jean-Sébastien Coron, Paul C. Kocher, and David Naccache. Statistics and secret leakage. In Yair Frankel, editor, *Financial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000, Proceedings*,

- volume 1962 of *Lecture Notes in Computer Science*, pages 157–173. Springer, 2000.
- [24] Guillaume Dabosville, Julien Doget, and Emmanuel Prouff. A new second-order side channel attack based on linear regression. *IEEE Trans. Computers*, 62(8):1629–1640, 2013.
- [25] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. Nessie proposal: the block cipher NOEKEON. Nessie submission, 2000. <http://gro.noekeon.org/>.
- [26] Joan Daemen and Vincent Rijmen. Rijndael for AES. In *AES Candidate Conference*, pages 343–348, 2000.
- [27] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001.
- [28] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.
- [29] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.
- [30] Orr Dunkelman, editor. *Topics in Cryptology - CT-RSA 2012 - The Cryptographers’ Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, volume 7178 of *LNCS*. Springer, 2012.
- [31] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 293–302. IEEE Computer Society, 2008.

- [32] Thomas Eisenbarth. *Cryptography and cryptanalysis for embedded systems*. PhD thesis, Ruhr University Bochum, 2010.
- [33] M. Abdelaziz Elaabid and Sylvain Guilley. Portability of templates. *J. Cryptographic Engineering*, 2(1):63–74, 2012.
- [34] J.H. Friedman and J.W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23(9):881–890, 1974.
- [35] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *LNCS*, pages 262–280. Springer, 2010.
- [36] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Oswald and Rohatgi [73], pages 426–442.
- [37] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. stochastic methods. In Goubin and Matsui [40], pages 15–29.
- [38] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side channel resistance validation. NIST non-invasive attack testing workshop, 2011. http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/08_Goodwill.pdf.
- [39] Louis Goubin and Ange Martinelli. Protecting AES with Shamir’s secret sharing scheme. In Preneel and Takagi [79], pages 79–94.
- [40] Louis Goubin and Mitsuru Matsui, editors. *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*. Springer, 2006.
- [41] Louis Goubin and Jacques Patarin. DES and differential power analysis (the ”duplication” method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES’99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

- [42] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. In Preneel and Takagi [79], pages 326–341.
- [43] Suvadeep Hajra and Debdeep Mukhopadhyay. On the optimal pre-processing for non-profiling differential power analysis. In Prouff [81], pages 161–178.
- [44] David Harris and Sarah Harris. *Digital Design and Computer Architecture, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2012.
- [45] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006.
- [46] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough - deriving optimal distinguishers from communication theory. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 55–74. Springer, 2014.
- [47] Holger H. Hoos and Thomas Stützle. *Stochastic local search: Foundations & applications*. Elsevier, 2004.
- [48] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.
- [49] John Kelsey, Bruce Schneier, and David Wagner. Key-schedule cryptanalysis of idea, g-des, gost, safer, and triple-des. In Koblitz [53], pages 237–251.
- [50] Stéphanie Kerckhof, François Durvaux, Cédric Hocquet, David Bol, and François-Xavier Standaert. Towards green cryptography: A comparison of lightweight ciphers from the energy viewpoint. In Prouff and Schaumont [85], pages 390–407.
- [51] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, vol. IX, 1883.

- [52] Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011.
- [53] Neal Koblitz, editor. *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*. Springer, 1996.
- [54] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Koblitz [53], pages 104–113.
- [55] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Wiener [118], pages 388–397.
- [56] Matthew Kwan. Reducing the gate count of bitslice DES. *IACR Cryptology ePrint Archive*, 2000:51, 2000.
- [57] Jen-Fue Maa, Dennis K. Pearl, and Robert Bartoszyński. Reducing multidimensional two-sample data to one-dimensional inter-point comparisons. *The Annals of Statistics*, 24(3):pp. 1069–1074, 1996.
- [58] François Macé, François-Xavier Standaert, and Jean-Jacques Quisquater. Information theoretic evaluation of side-channel resistant logic styles. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *LNCS*, pages 427–442. Springer, 2007.
- [59] David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.
- [60] Houssem Maghrebi, Claude Carlet, Sylvain Guilley, and Jean-Luc Danger. Optimal first-order masking with linear and non-linear bijections. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT*, volume 7374 of *LNCS*, pages 360–377. Springer, 2012.
- [61] Houssem Maghrebi, Emmanuel Prouff, Sylvain Guilley, and Jean-Luc Danger. A first-order leak-free masking countermeasure. In Dunkelman [30], pages 156–170.
- [62] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *LNCS*, pages 222–235. Springer, 2004.

- [63] Stefan Mangard, editor. *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers*, volume 7771 of *Lecture Notes in Computer Science*. Springer, 2013.
- [64] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
- [65] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Information Security*, 5(2):100–110, 2011.
- [66] 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 Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 486–505. Springer, 2013.
- [67] Marcel Medwed, Christophe Petit, Francesco Regazzoni, Mathieu Renauld, and François-Xavier Standaert. Fresh re-keying II: securing multiple parties against side-channel and fault attacks. In Prouff [80], pages 115–132.
- [68] Marcel Medwed and François-Xavier Standaert. Extractors against side-channel attacks: weak or strong? *J. Cryptographic Engineering*, 1(3):231–241, 2011.
- [69] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.
- [70] Amir Moradi and François-Xavier Standaert. Moments-correlating DPA. *IACR Cryptology ePrint Archive*, 2014:409, 2014.

- [71] David Oswald and Christof Paar. Improving side-channel analysis with optimal linear transforms. In Mangard [63], pages 219–233.
- [72] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical second-order DPA attacks for masked smart card implementations of block ciphers. In Pointcheval [78], pages 192–207.
- [73] Elisabeth Oswald and Pankaj Rohatgi, editors. *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*. Springer, 2008.
- [74] Bruno Paillard. An introduction to digital signal processors. <http://dsp-book.narod.ru/InDSPrcs.pdf>, 2002.
- [75] Telecom ParisTech. DPA contest v2. <http://www.dpacontest.org/v2/>, 2010.
- [76] Telecom ParisTech. DPA contest v3. <http://www.dpacontest.org/v3/>, 2012.
- [77] Bertram Poettering. AVRAES: The AES block cipher on AVR controllers. <http://point-at-infinity.org/avraes/>, 2007.
- [78] David Pointcheval, editor. *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*. Springer, 2006.
- [79] Bart Preneel and Tsuyoshi Takagi, editors. *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*. Springer, 2011.
- [80] Emmanuel Prouff, editor. *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, volume 7079 of *Lecture Notes in Computer Science*. Springer, 2011.
- [81] Emmanuel Prouff, editor. *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014,*

- Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*. Springer, 2014.
- [82] Emmanuel Prouff and Matthieu Rivain. A generic method for secure sbox implementation. In Seun Kim, Moti Yung, and Hyung-Woo Lee, editors, *Information Security Applications, 8th International Workshop, WISA 2007, Jeju Island, Korea, August 27-29, 2007, Revised Selected Papers*, volume 4867 of *Lecture Notes in Computer Science*, pages 227–244. Springer, 2007.
 - [83] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.
 - [84] Emmanuel Prouff and Thomas Roche. Attack on a higher-order masking of the AES based on homographic functions. In Guang Gong and Kishan Chand Gupta, editors, *INDOCRYPT*, volume 6498 of *LNCS*, pages 262–281. Springer, 2010.
 - [85] Emmanuel Prouff and Patrick Schaumont, editors. *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*. Springer, 2012.
 - [86] Ronald Pyke. Spacings revisited. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability (Univ. California, Berkeley, Calif., 1970/1971), Vol. I: Theory of statistics*, pages 417–427, Berkeley, Calif., 1972. Univ. California Press.
 - [87] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.
 - [88] Francesco Regazzoni, Alessandro Cevrero, François-Xavier Standaert, Stéphane Badel, Theo Kluter, Philip Brisk, Yusuf Leblebici, and Paolo Ienne. A design flow and evaluation framework for DPA-resistant instruction set extensions. In Clavier and Gaj [20], pages 205–219.

- [89] Mathieu Renauld, Dina Kamel, François-Xavier Standaert, and Denis Flandre. Information theoretic and security analysis of a 65-nanometer DDSLL AES S-box. In Preneel and Takagi [79], pages 223–239.
- [90] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *LNCS*, pages 109–128. Springer, 2011.
- [91] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting time samples for multivariate DPA attacks. In Prouff and Schaumont [85], pages 155–174.
- [92] Vincent Rijmen. *Cryptanalysis and Design of Iterated Block Ciphers*. PhD thesis, Katholieke Universiteit Leuven, 1997.
- [93] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
- [94] Matthieu Rivain, Emmanuel Prouff, and Julien Doget. Higher-order masking and shuffling for software implementations of block ciphers. In Clavier and Gaj [20], pages 171–188.
- [95] Thomas Roche and Emmanuel Prouff. Higher-order glitch free implementation of the AES using secure multi-party computation protocols - extended version. *J. Cryptographic Engineering*, 2(2):111–127, 2012.
- [96] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.
- [97] Kai Schramm and Christof Paar. Higher order masking of the AES. In Pointcheval [78], pages 208–225.

- [98] Bernard W. Silverman. *Density estimation for statistics and data analysis*. Monographs on Statistics and Applied Probability. Taylor & Francis, 1986.
- [99] François-Xavier Standaert, François Koeune, and Werner Schindler. How to compare profiled side-channel attacks? In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS*, volume 5536 of *LNCS*, pages 485–498, 2009.
- [100] François-Xavier Standaert, Eric Peeters, Cédric Archambeau, and Jean-Jacques Quisquater. Towards security limits in side-channel attacks. In Goubin and Matsui [40], pages 30–45.
- [101] François-Xavier Standaert, Christophe Petit, and Nicolas Veyrat-Charvillon. Masking with randomized look up tables - towards preventing side-channel attacks of all orders. In David Naccache, editor, *Cryptography and Security*, volume 6805 of *LNCS*, pages 283–299. Springer, 2012.
- [102] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 112–129. Springer, 2010.
- [103] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In Oswald and Rohatgi [73], pages 411–425.
- [104] 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 Pil Joong Lee and Jung Hee Cheon, editors, *Information Security and Cryptology - ICISC 2008, 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers*, volume 5461 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 2008.
- [105] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the*

Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

- [106] Francois-Xavier Standaert, Tal G. Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks (extended version). Cryptology ePrint Archive, Report 2006/139, 2006. <http://eprint.iacr.org/>.
- [107] François-Xavier Standaert, Eric Peeters, Gaël Rouvroy, and Jean-Jacques Quisquater. An overview of power analysis attacks against field programmable gate arrays. *Proceedings of the IEEE*, 94(2):383–394, 2006.
- [108] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013.
- [109] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Solid-State Circuits Conference, 2002. ESSCIRC 2002. Proceedings of the 28th European*, pages 403–406, Sept 2002.
- [110] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*, pages 246–251. IEEE Computer Society, 2004.
- [111] Michael Tunstall and Olivier Benoît. Efficient use of random delays in embedded software. In Damien Sauveron, Constantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems, First IFIP TC6 / WG 8.8 / WG 11.2 International Workshop, WISTP 2007, Heraklion, Crete, Greece, May 9-11, 2007, Proceedings*, volume 4462 of *Lecture Notes in Computer Science*, pages 27–38. Springer, 2007.
- [112] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In

- Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *LNCS*, pages 126–141. Springer, 2013.
- [113] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Wang and Sako [115], pages 740–757.
- [114] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Generic side-channel distinguishers: Improvements and limitations. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *LNCS*, pages 354–372. Springer, 2011.
- [115] Xiaoyun Wang and Kazue Sako, editors. *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *LNCS*. Springer, 2012.
- [116] Bernard L. Welch. The generalization of Student’s problem when several different population variances are involved. *Biometrika*, pages 28–35, 1947.
- [117] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. The myth of generic dpa...and the magic of learning. In Josh Benaloh, editor, *Topics in Cryptology - CT-RSA 2014 - The Cryptographer’s Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, volume 8366 of *Lecture Notes in Computer Science*, pages 183–205. Springer, 2014.
- [118] Michael J. Wiener, editor. *Advances in Cryptology - CRYPTO ’99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.