UNIVERSITY NAME

DOCTORAL THESIS

Thesis Title

Author:
John SMITH

Supervisor:
Dr. James SMITH

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Research Group Name
Department or School Name

July 18, 2018

Contents

List of Figures

List of Tables

List of Abbreviations

SCA	**S**ide **C**hannel **A**ttack
AES	**A**dvanced **E**ncryption **S**tandard
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
ANSSI	**A**gence **N**ational de la **S**écurité des **S**ystèmes d' **I**nformation
ITSEF	**I**nformation **T**echnology **S**ecurity **E**valuation **F**acility
CESTI	**C**entre d'**E**valuation de la **S**écurité des **T**échnologies de l'**I**nformation
EAL	**E**valuation **A**ssurance **L**evels
TOE	**T**arget **O**f **E**valuation
SFR	**S**ecurity **F**unctional **R**equirements
SAR	**S**ecurity **A**ssurance **R**equirements
CC	**C**ommon **C**riteria
ETR	**E**valuation **T**echnical **R**apport
EGV	**E**xplained **G**lobal **Variance**
ELV	**E**xplained **L**ocal **Variance**
PCA	**P**rincipal **C**omponents **A**nalysis
LDA	**L**inear **D**iscriminant **A**nalysis
KDA	**K**ernel Fisher **D**iscriminant **A**nalysis
SSS	**S**mall **S**ample **S**ize problem
NN	**N**eural **N**etwork
CNN	**Convolutional** **N**eural **N**etwork
DA	**D**ata **A**ugmentation

List of Symbols

$GF(2^b)$	Galois Field of order 2^b
$\mathbb{Z}_N^*$	...

Part I

Context and State of the Art

Chapter 1

Introduction

1.1 Introduction to Cryptography

The terms *Cryptography*, from the Greek *kryptòs* (secret) and *graphein* (writing), and *Cryptanalysis*, denote two branches of a science named *Cryptology*, or *science of the secret*. Cryptography initially refers to the art of *encrypting* messages, which means writing meaningful messages in such a way to appear nonsense to anyone unaware of the encryption process. The readable message is referred to as *plaintext*, while the unintelligible output of the encryption is referred to as *ciphertext*. In general, cryptography aims to construct protocols to secure communication, while cryptanalysis studies the resistance of cryptographic techniques, developing *attacks* to break the cryptosystems' security claims. These two complementary domains evolve in parallel, since the evolution of attack techniques allows conceiving more resistant cryptographic algorithms, and inversely the resistance of such algorithms requires the conception of more sophisticated attacks.

The art of cryptography is very ancient, probably as ancient as the language, but only the development of information technology made cryptology take the shape of a proper science, sometimes referred to as *Modern Cryptology*. The last is seen as a branch of different disciplines, such as applied mathematics, computer science, electrical engineering, and communication science. Modern cryptosystems exploit algorithms based on mathematical tools and are implemented as computer programs, or electronic circuits. Their goal is to provide security functionalities for communications that use *insecure channels*, for example the internet. In particular, modern cryptosystems are designed in order to ensure at least one of the four following information security properties:

a. *confidentiality*: the transmitted message must be readable only by a chosen pool of authorized entities;

b. *authenticity*: the receiver can verify the identity of the sender of a message;

c. *non-repudiation*: the sender of a message cannot deny having sent the message afterwards;

d. *data integrity*: the receiver can be convinced that the message has not been corrupted during the transmission.

Two branches of cryptography may be distinguished: the *symmetric cryptography* and the *asymmetric cryptography*. The first one historically appeared before and is based on the hypothesis that the two communicating entities share a common secret, or secret key; for this reason this is also called *secret key cryptography*. The second one, introduced around 1970, allows any entity to encrypt a message in such a way

that only a unique chosen other entity could decrypt it; this is also called *public key cryptography*.

A general principle in cryptography, nowadays widely accepted by cryptography researchers, is the one given by Kerckhoff in 19th century: it states that cryptosystems should be secure even if everything about the system, except the key, is public knowledge. Following this principle, today many industrials and governmental agencies exploit for their security services cryptosystems based over standardised algorithms. Such algorithms are of public domain, thus have been tested and tried to be broken by a large amount of people, before, during and after the standardization process. Resistance to many attempts of attacks is actually the strengths of standard algorithms.

Low-level cryptographic routines, called *primitives*, are often used as building blocks to construct cryptographic protocols. We provide hereafter a description of a standard primitive, the symmetric AES, whose implementation will be the target of all experiments described in this thesis.

1.1.1 Description of AES

The *Advanced Encryption Standard* (AES) has been standardized in 2001 by the United States governmental agency *National Institute of Standards and Technology* (NIST), through the *Federal Information Processing Standards Publication 197* (FIPS PUB 197) [NIS]. It is a *block cipher*, meaning that the encryption and decryption of the AES are functions that take as input a string (respectively the plaintext or the ciphertext) of fixed length over the binary alphabet. Indeed, the AES operates on blocks of 128 bits.[1] There exist three versions of AES, characterized by the size of the used key: 128, 192 or 256 bits. The encryption is done by rounds. The number of executed rounds depends on the key size (10 rounds for 128 bits, 12 for 192 and 14 pour 256). The basic processing unit in AES algorithm is a byte. For AES internal operations, bytes are arranged on a two-dimensional array called the *state*, denoted s. Such a state has 4 rows and 4 columns, thus contains 16 bytes. The byte lying at the i-th row, j-th column of s will be denoted by $s_{i,j}$ for $i, j \in \{0, 1, 2, 3\}$. The 16 input bytes and the 16 output bytes are indexed column-wise as shown in Fig. 1.1. Each element $s_{i,j}$ of a state is mathematically seen as an element of the *Rjindael finite field*, defined as $GF(2^8) = \mathbb{Z}/2\mathbb{Z}[X]/P(X)$ where $P(X) = X^8 + X^4 + X^3 + X + 1$. Five functions are performed during the AES, named KeySchedule, AddRoundKey, SubBytes, ShiftRows and MixColumns. At high level the AES algorithm is described hereafter:

Key Expansion: derivation of round keys from secret key through the KeySchedule function

Round 0:

AddRoundKey

Rounds 1 to penultimate:

SubBytes

[1] When a block cipher is used to encrypt a plaintext of different size, the plaintext is chunked into blocks of the appropriate one, and each block is encrypted accordingly to a so-called *mode of operation*.

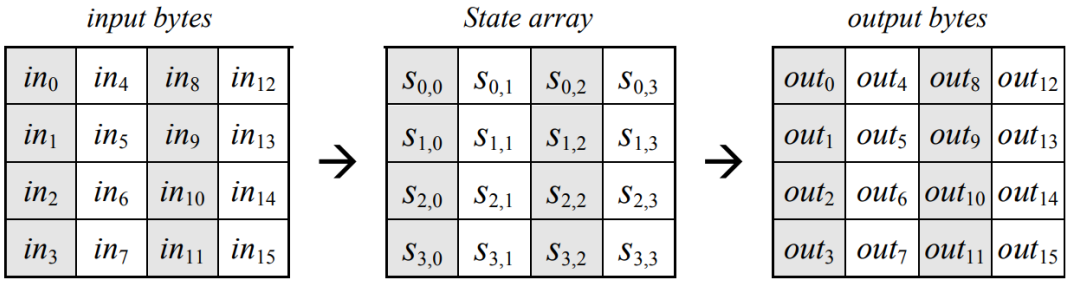


FIGURE 1.1: State array input and output. Source: [NIS].

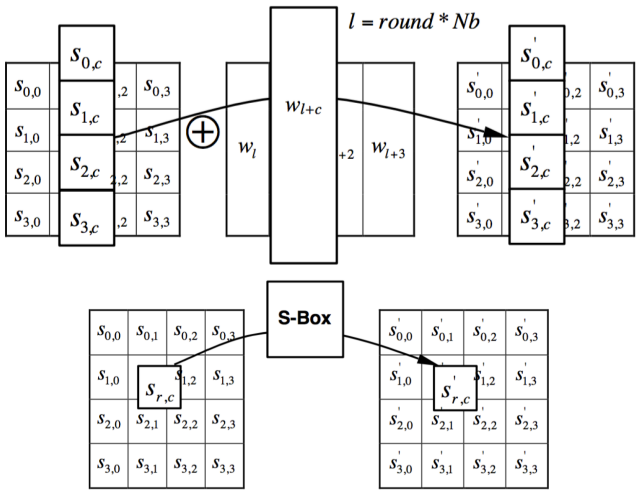


FIGURE 1.2: AddRoundKey (top) and SubBytes (bottom) operate over the State byte by byte, independently. Source: [NIS].

ShiftRows

MixColumns

AddRoundKey

Last Round:

SubBytes

ShiftRows

AddRoundKey

A description of the five functions is provided hereafter.

AddRoundKey

Each byte of the state is combined with the corresponding byte of the round key *via* an addition over the Rjindael field $GF(2^8)$, *i.e.* a bitwise exclusive OR (*XOR*) operation $\oplus$.

SubBytes

The SubBytes transformation is a non-linear byte invertible substitution that operates independently on each byte of the State using a substitution table (called Sbox). The SubBytes is composed of the following two functions:

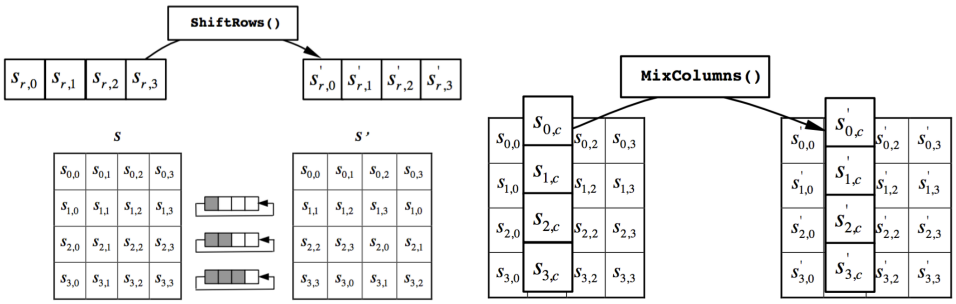


FIGURE 1.3: ShiftRows operates over the State rows. MixColumns operates over the State columns. Source: [NIS].

- the inversion in $GF(2^8)$ where the element $\{00\}$ is mapped to itself
- the affine transformation which maps each byte b_i to:

$$b_i \oplus b_{(i+4)\text{mod }8} \oplus b_{(i+5)\text{mod }8} \oplus b_{(i+6)\text{mod }8} \oplus b_{(i+7)\text{mod }8} \oplus c_i \,, \tag{1.1}$$

where c_i is the i^{th} bit of $\{63\} = (01100011)_2$.

ShiftRows

The bytes in the last second, third and fourth rows of the State are cyclically shifted over 1, 2, and 3 byte(s) respectively.

MixColumns

Each column of the State is treated as a four-term polynomial. They are considered as polynomials over the Rjindael field $GF(2^8)$ and multiplied modulo $X^4 + 1$ with a fixed polynomial $a(X) = \{03\}X^3 + \{01\}X^2 + \{01\}X + \{02\}$.

KeySchedule

To lighten notations, the KeySchedule is described for the 128-bits cipher, fixing many constants to the value 4. For the 192-bits and 256-bits such constants have to be fixed respectively to 6 and 8. The key round of the initial round of AES coincides with the secret encryption key $\boldsymbol{K} = (k_{0,0}, k_{0,1}, \ldots, k_{0,3}, k_{1,0}, \ldots, k_{1,3}, \ldots, k_{3,3})$. The i-th round key is given by

$$\boldsymbol{K_i} = (k_{4i,0}, k_{4i,1}, \ldots, k_{4i,3}, k_{4i+1,0}, \ldots, k_{4i+1,3}, \ldots, k_{4i+3,3}),$$

where $k_{4i+a,b}$ is calculated, for $i > 0$, $a \in \{0, \ldots 3\}$ and $b \in \{0, \ldots, 3\}$, as follows:

$$\begin{cases} k_{4i+a,b} = k_{4i+a-4,b} \oplus k_{4i+a-1,b} & \text{if } a \neq 0 \\ k_{4i+a,b} = k_{4i+a-4,b} \oplus \text{Sbox}(k_{4i+a-1,(b+1)\text{mod }4}) \oplus \text{Rcon}(a) & \text{if } a = 0 \text{ and } b = 0 \\ k_{4i+a,b} = k_{4i+a-4,b} \oplus \text{Sbox}(k_{4i+a-1,(b+1)\text{mod }4}) & \text{if } a = 0 \text{ and } b \neq 0 \,, \end{cases}$$

where $Rcon(a) = \{02\}^{a-1}$ in the Rjindael finite field,[2] and Sbox is the substitution table used for the SubBytes transformation.

[2] where $\{02\} = (00000010)_2$ is represented by the polynomial x

1.2 Secure Components

As we have seen in the previous section, modern cryptography proposes solutions to secure communications that ask for electronic computations and repose their security over some secret keys. Keys are represented as long bit strings, very hard to be memorized by users. Thus, keys need to be stored in a secure medium, and never delivered in clear over insecure channels. Smart cards were historically conceived as a practical solution to such a key storage issue: they consist in small devices a user can easily carry around with, which not only store secret keys, but also are able to internally perform cryptographic operations, in such a way that keys are never delivered. The registrations of a first patent describing memory cards by Roland Moreno in 1974 [Mor74], and of a second one describing cards equipped with microcontrollers by Michel Ugon in 1977 [Ugo77] are often referred to date the smart card invention, finally produced for the first time in 1979. Smart cards are pocket-sized plastic-made cards equipped with a secure component, which is typically an integrated circuit containing some computational units and some memories.

Today, about 40 years after its invention, they still have a huge diffusion, both in terms of applicative domains and in terms of quantity of exemplars. Indeed, they serve as credit or ATM cards, healthy cards, ID cards, public transport payment cards, fuel cards, identification and access badges, authorization cards for pay television, etc. Slightly changing the card support, we find other applications of the same kind of integrated circuits, for example the mobile phone SIMs (*Subscriber Identity Module*) and the electronic passports. In terms of quantity, a marketing research [Abi] found out that in 2014 8.8 billion smart cards have been sold, *i.e.* the same order of magnitude of the global population.

In addition to smart cards, the recent growing and variation of security needs lead to the development and specification of other kinds of secure solutions, for example the *Trusted Platform Module* (TPM), which is a secure element providing cryptographic functionalities to a motherboard, or completely different solutions based on software layers, that are today in great expansions. An example is provided by the *Trusted Execution Environment* (TEE), which is a software environment of the main processor of a smartphone or tablet, designed to assure resistance to software menaces.

1.2.1 Embedded Cryptography Vulnerabilities

1.2.1.1 Side-Channel Attacks

Until the middle of the nineties, the security of embedded cryptosystems was considered, in the public domain, as equivalent to the mathematical security of the cryptographic algorithm. In classical cryptanalysis, an attacker usually has the knowledge of the algorithm (in accordance to Kerckhoff's principle) and of some inputs and/or outputs. Starting from these data, his goal is to retrieve the secret key. This attack model considers the algorithm computation as a black box, in the sense that no internal variable can be observed during execution, only inputs and/or outputs. With his seminal paper about Side-Channel Attacks (SCAs) in 1996, Paul Kocher showed that such a black-box model fails once the algorithm is implemented over a physical component [Koc96]: an attacker can indeed inspect its component during the execution of the cryptographic algorithm, monitor some physical quantities

TABLE 1.1: Classification of Harware Attacks

	Passive	Active
Invasive		
Semi-Invasive	(SCAs)	(FAs)
Non-Invasive	SCAs	FAs

(*e.g.* the execution time [Koc96] or the instantaneous power consumption [KJJ99]) and deduct information about internal variables of the algorithm. Depending on the attacked algorithm, making inference over some well chosen internal variables (the so-called *sensitive variables* of the algorithm) is sufficient to retrieve the secret key. After these first works it was shown that other observable physical quantities contained *leakages* on sensitive information; for example the electromagnetic radiation emanating from the device [GMO01; QS01] and the acoustic emanations [GST14]. Moreover, if until few years ago it was thought that only small devices, equipped with slow microprocessors and with small-sized architecture, such as smart cards, were vulnerable to this kind of Side-Channel Attacks, the last cited recent work about acoustic emanations, together with other works exploiting electromagnetic fluctuations pointed out that much faster and bigger devices, *i.e.* laptops and desktop computers, are vulnerable as well [Gen+15; GPT15; Gen+16].

1.2.1.2 A Classification of the Attacks against Secure Components

The Side-Channel Attacks outlined in previous paragraph, and which are the main concern of this thesis, belong to a much bigger family of hardware attacks that can be performed to break cryptographic devices security claims. A classification for hardware attacks is briefly outlined in Tab. 1.1. They are commonly classified on the base of two criteria: on one hand we can distinguish passive and active attacks, on the other hand we can distinguish invasive, semi-invasive, non-invasive attacks.

Passive attacks: in passive attack, the device run respecting its specifications. The attacker observes its behaviour without provoking any alteration;

Active attacks: in active attacks a special manipulation is performed in order to corrupt the normal behaviour of the device.

Invasive attacks: in invasive attacks, the device is unpackaged and inspected at the level of the component technology. The circuit can be modified/broken, signals can be accessed *via* a probing station, etc. There is no limits to the manipulations an attacker can do to the components;

Semi-invasive attacks: as in invasive attacks the device is unpackaged, but in contrast to them, no direct electrical contact to the chip is done;

Non-invasive attacks: in non-invasive attacks the device is not modified and only accessible interfaces are exploited.

In the literature, the term Side-Channel Attacks (SCAs) commonly refers to the passive non-invasive attacks. Nevertheless, the techniques proposed under the name of SCAs, that always require the acquisition of some signals, might also include attacks where the device is unpacked, in order to improve the signal amplitude. In this sense, SCAs belong to the semi-invasive group of attacks as

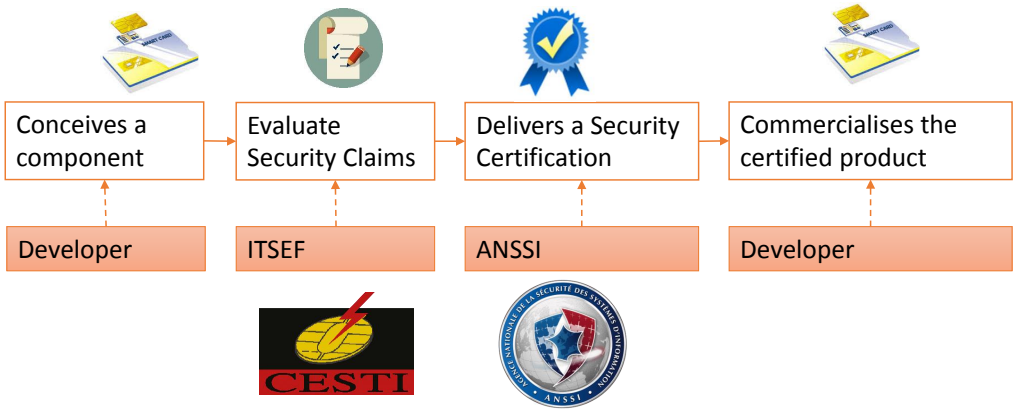


FIGURE 1.4: The actors of French Certification Scheme

well. Similarly, active non-invasive attacks are often referred to as *Fault Injection Attacks*, that might also be run in a semi-invasive way.

Beyond hardware attacks, there exists a second class of attacks that menaces the security of cryptographic devices: the software attacks. In contrast with hardware attacks, software attacks exploit vulnerabilities that are not related to the physical implementation of the cryptographic functionalities of the device: they are not based on hypotheses about the material execution of the cryptographic algorithms, but exploit vulnerabilities of the software interfaces. A typical example of software attack consists in charging malware code into the device, enabling access to data and instructions contained in memories (RAM or ROM), in order to retrieve, modify or destroy information they hold. In last years, together with the growing complexity of secure devices, attacks become more and more sophisticated and the boundary between hardware and software attack is more and more blurred. Moreover combined software/hardware attacks [].

1.2.2 Certification of a Secure Hardware - The Common Criteria

In previous paragraphs we have evoked the great diffusion of the cryptographic devices and the existence of a wide range of attacks exploiting vulnerabilities coming from the way cryptography is embedded. These two factors imply a great risk related to the production and commercialization of such devices, and justify the importance and necessity to ensure reliability on their security claims. This necessity lead to the arise of several guidelines and standards for their evaluation. The international standard ISO/IEC 15408, also known as *Common Criteria for Information Technology Security Evaluation* (abbreviated as *Common Criteria* or simply *CC*) represents one of the strongest efforts in standardization, unifying in 1999 three previously existing standards:

- the *Trusted Computer System Evaluation Criteria* (TCSEC - United States - 1983)
- the *Information Technology Security Evaluation Criteria* (ITSEC - France,Germany, Netherlands, United Kingdom - 1990)
- the *Canadian Trusted Computer Product Evaluation Criteria* (CTCPEC - Canada - 1993).

TABLE 1.2: Evaluation Assurance Levels

EAL	Description
EAL1	Functionally tested
EAL2	Structurally tested
EAL3	Methodically tested and checked
EAL4	Methodically designed, tested and reviewed
EAL5	Semi-formally designed and tested
EAL6	Semi-formally verified design and tested
EAL7	Formally verified design and tested

1.2.2.1 The actors

The CC define four actors of the evaluation process of a secure component:

- **The Developer**, who conceives a product and wishes to sell it as a certified secure product. He sends a request for evaluation to the certification body and, once the request is accepted, he contacts an evaluation laboratory;
- **The ITSEF** is the *IT Security Evaluation Facility*; in France it is named *Centre d'Evaluation de la Securité des Technologies de l'Information* (CESTI). It is an evaluation laboratory, in possession of a certification body agreement, which performs the security tests to assess the resilience of the product;
- **The Certification Body** is often a governmental organism, the *Agence National de la Securité des Systèmes d'Information* (ANSSI) in France, or the *Bundesamt für Sicherheit in der Informationstechnik* (BSI) in Germany, for example. It ensures the quality of the evaluation and delivers a certificate to the developer;
- **The end user**, who buys the product and follows its security guidelines.

1.2.2.2 The Target of Evaluation and the security objectives

To start the certification process, the developer compiles a document called *Security Target* (ST). Such a document begins specifying the (part of the) device subjected to evaluation, the so-called *Target of Evaluation* (TOE), then lists its *Security Functional Requirements* (SFR), choosing among those proposed by the CC. In practice, and to ease the redaction of the ST, the choice of the SFRs is not open, but guided by the typology of the component. In particular, the CC propose a catalogue of *Protection Profiles*, associated with the required SFRs. For example *smart card* or *TEE* designate some precise Protection Profiles. They differ in various aspect, and their main difference is that until now, TEE are not required to be resistant to hardware attacks, but only software ones. The recent alerts pointed out in works such as those cited at the end of Sec. 1.2.1.1 lead us to think that such a restriction to sofware vulnerability analysis will soon be replaced by a larger requirement.

1.2.2.3 Evaluation Assurance Level and Security Assurance Requirements

In CC seven *Evaluation Assurance Level* (EAL) are defined. They determine the quantity and complexity of the tasks the evaluator has to effectuate, thus specifying the insurance strength. The EAL are defined in insurance increasing order, so that the

EAL1 has the lowest verification exigences while EAL7 has the highest ones. In Table 1.2 the objectives given by the CC for each EAL are resumed.

During the process of evaluation, the SFRs of the TOE have to be verified according to the claimed EAL. To this end, the evaluation is divided into six classes of *Security Assurance Requirement* (SAR). Five of this classes are the so-called *conformity* classes, and one is the *vulnerability assessment* class. Each class is sub-divided in several *families* (excepted the vulnerability assessment class, which only contains one family), and the evaluators are charged to check each requirement corresponding to these families. The Table 1.3 resumes the SAR classes and their families. For each family a grade is assigned following precise specifications detailed in CC, and the obtention of a certain EAL depends on the grades obtained for each family, as reported in Table 1.4. An EAL can also be *augmented*, meaning that the product achieves all the required SAR grades to obtain a certain EAL and some upper grades for certain families. For example, smart cards are usually protected at level EAL4+AVA_VAN5+ALC_DVS2, and chips for e-passport application are usually protected at level EAL5+AVA_VAN5+ALC_DVS2. In case of banking smart cards, the card also needs to respect the EMVco norms, being EMVco a consortium of six companies (Visa, MasterCard, JCB, American Express, China UnionPay, and Discover) that manages private certification schemes for banking cards, payment terminal and automated teller machines.

1.2.2.4 The AVA_VAN family and the Attack Potential

The AVA_VAN is the solely family of the vulnerability assessment SAR. The goal of such a SAR is to make the connection between the conformity of the TOE, verified *via* the analysis of its documentation, and the efficiency of its protections and countermeasures. This is the step of the evaluation in which the actual resilience of the TOE against the *penetration tests* is measured. In this phase the attacks outlined in Sec. 1.2.1 are taken into account, and the so-called *attack potential* of such attacks is stated. The attack potential is a notion appearing in CC whose aim is to reflect the realism of succeeding a certain attack, and thus its realistic dangerousness. Indeed in the context of physical attacks, many possible attack paths require unrealistic conditions, amounts of time and/or money to be actually performed on the field and do not represent in reality a great risk. For example, invasive attacks such as probing attacks which appears in theory the most dangerous ones, ask in general for some very expensive instruments, a huge expertise, much time and many broken samples before succeeding. Their attack potential can thus result not so wondering. For this evaluation phase, the evaluator is in charge to prepare a testing plan. This is a list of the possibly dangerous attack paths, basing on a code analysis, and on the state-of-the-art attacks list in general provided by working groups dedicated to the secure component considered. Once the testing plan is ready he practically tests each attack. For each succeeded attack he fills a *cotation table* in order to assign a score to the attack, on the basis of several criteria. The goal of the cotation table is to provide a metric enabling to compare very different kinds of attacks. The guidelines for the cotation table are given by the *Common Methodology for Information Technology Security Evaluation* (CEM).

In the case of smart cards, the evaluation systematically includes the AVA_VAN5

TABLE 1.3: Security Assurance Requirements

Class	Family	Description
Development	ADV_ARC	Security architecture
	ADV_FSP	Functional specification
	ADV_IMP	Implementation representation
	ADV_INT	TOE Security Functions internals
	ADV_SPM	Security policy modelling
	ADV_TDS	TOE design
Guidance Documents	AGD_OPE	Operational user guidance
	AGD_PRE	Preparative procedures
Life-cycle support	ALC_CMC	Configuration Management capabilities
	ALC_CMS	Configuration Management scope
	ALC_DEL	Delivery
	ALC_DVS	Development security
	ALC_FLR	Flaw remediation
	ALC_LCD	Life-cycle definition
	ALC_TAT	Tools and techniques
ST evaluation	ASE_CCL	Conformance claims
	ASE_ECD	Extended components definition
	ASE_INT	ST introduction
	ASE_OBJ	Security objectives
	ASE_REQ	Security requirements
	ASE_SPD	Security problem definition
	ASE_TSS	TOE summary specification
Tests	ATE_COV	Coverage
	ATE_DPT	Depth
	ATE_FUN	Functional tests
	ATE_IND	Independent testing
Vulnerability assessment	AVA_VAN	Vulnerability analysis

TABLE 1.4: Required grades for the obtention of each EAL.

Family	Assurance Components by EAL						
	EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
ADV_ARC		1	1	1	1	1	1
ADV_FSP	1	2	3	4	5	5	6
ADV_IMP				1	1	2	2
ADV_INT					2	3	3
ADV_SPM						1	1
ADV_TDS		1	2	3	4	5	6
AGD_OPE	1	1	1	1	1	1	1
AGD_PRE	1	1	1	1	1	1	1
ALC_CMC	1	2	3	4	4	5	5
ALC_CMS	1	2	3	4	5	5	5
ALC_DEL		1	1	1	1	1	1
ALC_DVS			1	1	1	2	2
ALC_FLR							
ALC_LCD			1	1	1	1	2
ALC_TAT				1	2	3	3
ASE_CCL	1	1	1	1	1	1	1
ASE_ECD	1	1	1	1	1	1	1
ASE_INT	1	1	1	1	1	1	1
ASE_OBJ	1	2	2	2	2	2	2
ASE_REQ	1	2	2	2	2	2	2
ASE_SPD		1	1	1	1	1	1
ASE_TSS	1	1	1	1	1	1	1
ATE_COV		1	2	2	2	3	3
ATE_DPT			1	1	3	3	4
ATE_FUN		1	1	1	1	2	2
ATE_IND	1	2	2	2	2	2	3
AVA_VAN	1	2	2	3	4	5	5

grade, thus the testing plan is asked to be as complete as possible. The state-of-the-art of the attacks is periodically upgraded by the *JIL*[3] *Hardware Attacks Subgroup* (JHAS), a subgroup of the working committee *Senior Officials Group Information Systems Security* (SOG-IS) which coordinates the standardization of CC. Moreover, the JHAS produces the *Application of Attack Potential to Smartcards* [Lib13] of the JIL, which is an interpretation of the CEM in the special case of smart cards. The cotation table factors specified by the JHAS are detailed in Table 1.5. The evaluation is divided in two parts, an *identification* part, that reflects the difficulty in finding the attack path, and an *exploitation* part, that reflects the difficulty in actually performing the attack. The total score of an attack is the sum of scores assigned to each factor. To obtain the AVA_VAN5 grade every successful attack tested by the evaluators must have been rated at least 31.

1.2.2.5 The Evaluation Technical Report

The evaluation ends with the redaction by the evaluators of an *Evaluation Technical Report* (ETR), which is transmitted to the certification body. It analyses the ETR and, if the security claims of the TOE are verified, issues a *certificate*. The ETR is kept confidential. Concerning the penetration testing of a certified smart card, the ETR contains all the cotation tables of the succeeded attacks. If the component is certified it means that the score of those attacks was higher than 31, and such vulnerabilities are kept as *residual vulnerabilities*. The ETR is strictly reviewed annually by the evaluators in charge of the surveillance of the certificate. For the penetration testing, the evaluators are in particular asked each year to verify that the cotation of the attacks presented in the ETR did not drop.

1.3 This thesis objectives and contributions

Among the factors observable in the cotation table 1.5 we find *open samples*, interpretable as *device with known secrets*. Indeed, for an evaluation scope it is sometimes possible for an ITSEF to have access to a device identical to the TOE but where the evaluator can fix or access certain variables, for example some random numbers used by cryptographic algorithm, or load specific software. An evaluator may use this possibility in order to launch executions in which he is aware of the complete execution flow, including operations, manipulated internal variables (internally generated random ones as well) and register accesses. In this way he can understand and characterise the relations between the internal behaviour of the device and the physical observations, before performing a proper attack.

In the context of Side-Channel Attacks, when such a characterization phase is possible we talk about *profiling attacks*. Due to the favourable condition of this attacks, they are commonly considered the most dangerous ones, allowing a sort of worst-case security analysis. This thesis is mainly focused over such a profiling scenario. Indeed, we will address the problems an evaluator deals with when he is in such a favourable case and he wonders how to optimally exploit such a characterization phase in order to be able in the proper exploitation phase to extract as much information as possible from his signals. One of these issues is the selection

[3] Joint Interpretation Library

TABLE 1.5: Factors of the *Attack Potentials for Smartcards*

Factors	Identification	Exploitation
Elapsed Time		
<one hour	0	0
<one day	1	3
<one week	2	4
<one month	3	6
>one month	5	8
Expertise		
Layman	0	0
Proficient	2	2
Expert	5	4
Multiple Expert	7	6
Knowledge of the TOE		
Public	0	0
Restricted	2	2
Sensitive	4	3
Critical	6	5
Very critical hardware design	9	NA
Access to TOE		
<10 samples	0	0
<30 samples	1	2
<100 samples	2	4
>100 samples	3	6
Equipement		
None	0	0
Standard	1	2
Specialized	3	4
Bespoke	5	6
Multiple Bespoke	7	8
Open Samples		
Public	0	NA
Restricted	2	NA
Sensitive	4	NA
Critical	6	NA

of the so-called *Points of Interest* (PoI), strictly linked to the more general problem of dimensionality reduction.

1.3.1 Foreword of this Thesis: Research of Points of Interest

To perform a Side-Channel Attack, the monitoring of unintentional channels leaking from the attacked device is usually performed through an oscilloscope that samples continuous analog signals and turns them into discrete digitalized sequences. Such sequences are often referred to as *traces*. To allow a deep inspection of the device, the sampling rate of the oscilloscope needs to be high, leading very often to a high dimensionality of such traces. Nevertheless, it is expected that only a limited number of time samples are relevant for a SCA: those that are statistically dependent on the sensitive variable that is exploited to run the attack. Such time samples are called *Points of Interest* (PoIs). In the literature a few of different statistics was proposed and exploited to select such PoIs in a preliminary attack phase, in order to reduce both time and memory complexity of the attacks. A brief overview of such statistics is proposed in Sec. 2.2.4. The foreword of this thesis was to propose new methods to research and characterise the PoIs, in order to ameliorate and possibly optimise the preliminary attack phase consisting in their selection.

1.3.2 Dimensionality Reduction Approach

Beyond the use of point-wise statistics to identify the PoIs, an axe of research was launched in SCA context, importing from the Machine Learning domain more general techniques for dimensionality reduction of data. Around 2014, linear methods were drawing a raising attention, consisting in techniques to conveniently exploit linear combinations of many time samples. The first contributions we proposed belong to this axe of research: in Chapter 4 we describe the two mainly deployed techniques, the Principal Component Analysis and the Linear Discriminant Analysis, and tackle some open issues about their application to SCA context. The solutions proposed in the thesis have been presented at CARDIS 2015 [**cagli2015enhancing**] and published in the proceedings of this international conference.

Nowadays every device needing to obtain an AVA_VAN5 grade is equipped of specific countermeasures against SCAs. A brief overview of some classic and public principles providing efficient countermeasure is provided in Sec. 2.3. Among them, the *masking*, or *sharing*, countermeasures may be considered the most effective ones. Beyond the formal proofs of their efficiency provided in the literature [ISW03; PR13; Bar+15], they are the ones that most likely require a strong adaptation of the attack strategy in order to be defeated. Indeed, when an effective masking scheme is implemented, each sensitive variable of the original computation is split into shares randomly drawn, in such a way that any proper subset of shares is statistically independent of the sensitive variable itself. Computation of cryptographic primitives is done accessing only the random shares, with intermediate steps computing only the shares of the result. This forces the attacker to work with the joint distributions of the signal at the time samples where the shares are being accessed. In other words, point-wise statistics to retrieve PoIs are completely inefficient in presence of a masking countermeasure, since each time sample is by itself statistically independent from any sensitive variable. Moreover, interesting joint distributions have to be studied at their higher-order statistical moments to retrieve sensitive-data dependencies, implying that any linear method to combine time samples is inefficient as

well. To sum up, the issue of selecting PoIs or applying dimensionality reduction to side-channel traces protected by masking presents challenging difficulties. Such a hardness is mitigated when the attacker is able to perform a profiling phase during which he has the knowledge of the random values assigned to the shares during execution. In practice it is not always the case, so in this thesis we tackle the issue when such knowledge is absent, and we propose in Chapter 5, on the basis of a work presented at CARDIS 2016 [CDP16b], to deploy Kernel Fisher Discriminant Analysis (KDA) as a solution. This is an extension of the LDA dimensionality reduction technique, allowing applying some strategy to non-linearly combine time samples.

1.3.3 Towards Machine Learning and Neural Networks Approach

As a general observation about the track we followed during this thesis, we started from the problem of identifying the PoIs in a signal, that is classically tackled by means of pure statistical tools, such as hypothesis tests, then enlarged both the objectives and the methodologies. Indeed we observed that what mainly influences the successfulness of a Side-Channel Attack is the quality of the way information is extracted from data. Extracting information concerns approximating probability distributions that allow distinguishing different secret values. The first SCAs proposed in the literature acted in a point-wise fashion, *i.e.* were related to data distributions in single time samples of the acquisitions. In this sense the selection of such time samples, the PoIs, played a fundamental role and were a preliminary objective of these researches. As soon as one steps back to the final objective, *i.e.* defining and well approximating distinguishable distributions, the fact of completely discard a great part of time samples, selecting only a few of them, seems a waste. Convenient ways to combine time samples might turn into some resulting features whose distributions might have a greater distinguishability. This observation lead to a one-step back objective: determine such convenient ways. In this sense we explored feature extraction (or dimensionality reduction) tools, in order to preprocess data and turn rough data into compact ones whose distributions were distinguishable. Linear tools were analysed in a first time (PCA and LDA in particular), then non-linear tools (the KDA) were investigated to satisfy a necessary condition in order to deal with masked implementations.

Aware of the fact the the just cited tools are in the middle ground between classical multivariate statistics and the Machine Learning domain, we started exploring such a domain, that is today in fast development. The wide interest for Machine Learning is today justified by the trend of sense and analyse data of huge dimension for an always increasing variety of applications. To do so, more and more complex models have been explored, too complex to be treated with a formal statistical asset. The Machine Learning asset carries with him some intrinsic non-optimality, formalised by the so-called *No Free Lunch theorem*, briefly stated in Sec. 3.1.6, but is today demonstrating its capacities. We observe that Side-Channel Attacks belong to the kind of applications that might take advantage of Machine Learning tools, since they act by sensing and analysing data of high dimension. For this reason, in last years, a transfer from Machine Learning to the application domain of SCA started, and our researches make parts of such a flow.

The study of nowadays privileged tools in Machine Learning allowed us making a further step back toward the SCAs objective. Instead of look for a convenient preprocessing of data, whose output distributions have discriminant abilities, we

switched to look for models that directly approximate the distributions from rough data. This approach is proper to a branch of Machine Learning, called Deep Learning. The Deep Learning paradigm suggests to integrate the whole learning phase (in our case the whole processing leading to the discriminant distributions approximation) in a unique process, integrating in it any preprocessing. This is done considering multi-layered models, in particular Neural Networks, on which we finally focused. They are non-linear models, implying that they are able to eventually deal with side-channel traces protected by masking countermeasure. Moreover, some special structures of Neural Networks, the so-called Convolutional Neural Networks (CNNs), originally conceived for image recognition application, fit well to handle other kinds of classic countermeasures: those improving trace desynchronization, or misalignment (see Sec. 2.3). Classically, misalignment had to be treated by other special preprocessing phases, that Deep Learning paradigm allows to integrate in the single learning process. In Chapter 6, on the basis of the publication presented at CHES 2017 [CDP17], we discuss about the advantages of exploiting such CNNs in SCA context.

Beyond the application of the CNNs we discuss in Chapter 6, we believe that many kinds of side-channel scenarios, and especially profiling contexts, may be turned into a Machine Learning language and many researches already carried out for other applications should be exploited to understand if they represent or not a danger in embedded security domain, leading to powerful Side-Channel Attacks.

The next two chapters aim to briefly introduce preliminaries about these two vast domains: in Chapter 2 basic concepts about Side-Channel Attacks are provided, while Chapter 3 introduces some notions of Machine Learning.

Chapter 2

Introduction to Side-Channel Attacks

2.1 Notations and Probability and Statistics Recalls

Basic Notations. In this thesis we use calligraphic letters as $\mathcal{X}$ to denote sets, the corresponding upper-case letter X to denote random variables (random vectors $\vec{X}$ if with an arrow) over $\mathcal{X}$, and the corresponding lower-case letter x (resp. $\vec{x}$ for vectors) to denote realizations of X (resp. $\vec{X}$). The cardinality of a set $\mathcal{X}$ is denoted by $|\mathcal{X}|$. Matrices will be denoted with bold capital letters. The i-th entry of a vector $\vec{x}$ is denoted by $\vec{x}[i]$, while the transposed of a vector $\vec{x}$ is denoted as $\vec{x}^\intercal$. In general the ith observation of a random vector X will be denoted by x_i. Observations will sometimes be grouped into datasets $\mathcal{D} = \{x_1, \ldots, x_N\}$. Throughout this thesis, a finite set $\mathcal{Z} = \{z^1, \ldots, z^{|\mathcal{Z}|}\}$ will be often considered: it will always denote the possible values for a *sensitive variable Z* (see later). Its elements are ordered by overwritten indeces $z^1, z^2, \ldots$ and are sometimes encoded via a so-called *one-hot-encoding*: to each element z^j a $|\mathcal{Z}|$-dimensional vector $\vec{z}^j$ is associated, with all entries equal to 0 and the j-th entry equal to 1: $z^j \to \vec{z}^j = (0, \ldots, 0, \underbrace{1}_{j}, 0, \ldots, 0)$.

Probability Notations. The probability of a random variable X taking value in a subset $\mathcal{U} \in \mathcal{X}$ is denoted by $\mathrm{Prob}(X \in \mathcal{U})$, or simply by $\mathrm{Prob}\mathcal{U}$ if not ambiguous When $\mathcal{U}$ is reduced to a singleton $\mathcal{U} = x$ the same probability is denoted by $\mathrm{Prob}(X = x)$ or simply by $\mathrm{Prob}x$ if not ambiguous. If X is a discrete variable p_X denotes its probability mass function, such that $\mathrm{Prob}(X \in \mathcal{U}) = \sum_{x \in \mathcal{U}} p_X(x)$. The same symbol p_X is used to denote the probability density function (pdf for short) if X is a continuous variable, such that $\mathrm{Prob}(X \in \mathcal{U}) = \int_{\mathcal{U}} p_X(x)dx$, for $[a, b] \subset \mathcal{X}$. The symbol $\mathbb{E}[f(X)]$, or equivalently $\mathbb{E}_X[f(X)]$, denotes the expected value of a function f of the random value X, under the distribution of X. In the same way, symbols $\mathrm{Var}(X)$ and $\mathrm{Var}_X(X)$ denote the variance of X.

When two random variables X and Y are considered, their joint probability is denoted by $\mathrm{Prob}(X = x, Y = x)$, or simply by $\mathrm{Prob}(x, y)$ if not ambiguous, and their joint probability density (or mass) function is denoted by $p_{X,Y}(x, y)$. The conditional probability of X assuming the value x given an outcome y for Y is denoted by $\mathrm{Prob}(X = x \mid Y = y)$, or simply by $\mathrm{Prob}(x \mid y)$ if not ambiguous. The conditional probability density (or mass) function of X given an outcome y for Y is denoted by $p_{X \mid Y=y}(x)$. Finally, the covariance of the two variables is denoted by $\mathrm{Cov}(X, Y)$.

Bayes' Theorem. We recall some basic probability rules. For every $x \in \mathcal{X}$ and for every $y \in \mathcal{Y}$ we have what follows:

- *Symmetry of joint probabilities:* $p_{X,Y}(x,y) = p_{Y,X}(y,x)$;
- *Marginal probabilities from joint ones:* $p_X(x) = \sum_{Y=y} p_{X,Y}(x,y)$ (where the sum has to be intended as an integral if Y is a continuous variable);
- *Product rule:* $p_{X,Y}(x,y) = p_{Y \mid X=x}(y) p_X(x)$;

These rules are sufficient to demonstrate, in the case of discrete random variables X, Y, a key stone of probability theory, the Bayes' theorem:

$$p_{X \mid Y=y}(x) = \frac{p_{Y \mid X=x} p_X(x)}{p_Y(y)} \; ; \tag{2.1}$$

the marginal probability function p_X is referred to as *prior* probability of X, and describes the distribution of X without taking into account the variable Y. The conditional probability $p_{X \mid Y=y}$ is referred to as *posterior* probability of X, and gives the distribution of X once the outcome y of Y is taken into account. Notions of measure's theory are needed to show that Bayes' theorem is valid and keeps unchanged in case of continuous random variables and in cases in which one of the two involved variables is discrete and the other one is continuous. The interested reader might refer to [Fel08].

The Gaussian distribution. The Gaussian or normal distribution is a widely used model for the distribution of continuous variables. We use the symbol $X \sim \mathcal{N}(\mu, \sigma^2)$ to denote a random variable X that follows a Gaussian distribution with parameters $\mu \in \mathbb{R}$ and $\sigma^2 \in \mathbb{R}^+$. For a D-dimensional random vector $\vec{X}$ we use the symbol $X \sim \mathcal{N}(\vec{\mu}, \Sigma)$ to denote a vector that follows a multivariate Gaussian distribution with parameter $\vec{\mu} \in \mathbb{R}^D$ and $\Sigma \in \mathbb{R}^{D \times D}$ positive-definite. The density of the Gaussian distribution is completely determined by the value of its two parameters. It is given by the following expressions, in unidimensional and multidimensional case:

$$p_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2 , \tag{2.2}$$

$$p_{\vec{X}}(\vec{x}) = \frac{1}{\sqrt{2\pi \det(\Sigma)}} \exp -\frac{1}{2} (\vec{x} - \vec{\mu})^\intercal \Sigma^{-1} (\vec{x} - \vec{\mu}) \, . \tag{2.3}$$

The expected value of a Gaussian distribution coincides with the parameter μ for the univariate case and with $\vec{\mu}$ for the multivariate one. The parameter σ^2 coincides with the variance of the univariate distribution, while Σ coincides with the covariance matrix of the multivariate one.

Basics of Statistics. The word *statistics* refers to a branch of mathematics that aims to analyse, describe or interpret observed data. Differently, the word *statistic* refers to any measure obtained applying a function to some observed data. Let $\mathcal{D} = \{x_1, \ldots, x_N\}$ be a dataset of observations of a random variable X. We might distinguish two sub-branches in statistics: the descriptive statistics, and the inferential statistics. In descriptive statistics data are described by means of more or less complex statistics (in the sense of measure), the most common of them being the

arithmetic mean:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \,. \tag{2.4}$$

In inferential statistics data are considered as sample observation of random variables and the data analysis aims at modelling the distribution of such variables. Dealing with random variables, inferential statistics exploit the probability theory framework and theorems. Statistics of data (in the sense of measures) play an important role in inferential statistics as well, usually aiming to estimate some random variable parameters. In this case they are called *estimators* and will be denoted by a hat: for example, $\hat{\mathbb{E}}[X]$ denotes an estimator for the expected value of X and $\hat{\mathrm{Var}}(X)$ denotes an estimator for the variance of X. The most classical estimator for the expected value is the arithmetic mean $\overline{x}$. It has several valuable properties, for example it is *unbiased*, in the sense that, considering it as a random variable, its expected value coincides with the true value of $\mathbb{E}[X]$. Moreover, it is the *maximum-likelihood* estimator under the Gaussian distribution: for data that are independent among each other and drawn from a Gaussian distribution, the arithmetic mean of the observed data $\mathcal{D}$ is the value for the parameter μ that maximises the probability of observing the data $\mathcal{D}$. A common unbiased estimator for the variance is the following so-called sample variance:

$$\hat{\mathrm{Var}} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2 \,; \tag{2.5}$$

when the observed random variable follows a Gaussian distribution, such an estimator differs from the maximum-likelihood one, in which the factor $\frac{1}{N-1}$ is substituted by $\frac{1}{N}$. In the same way, acting with Gaussian random vectors, the maximum-likelihood estimator of the covariance matrix is biased, and differs from the common unbiased one for a multiplicative factor.

Various approaches exist to make statistical inference. The two main ones are the frequentist approach and the Bayesian one. The frequentist inference is an approach that draws conclusions exclusively from sample data. It makes use of methodologies like the statistical hypothesis testing and the confidence interval, and in this thesis it is sometimes referred to as classical statistics. In the frequentist approach, parameters that define the distribution of the analysed random variable are priorly considered as fixed and unknown, and are estimated or tested on the sole basis of the observation of the sample data $\mathcal{D}$. A second approach is the Bayesian inference, for which parameters that describe the analysed random variable are admitted to be probabilistic: in Bayesian inference, before the observation of sample data, the parameters have a prior distribution that reflects the knowledge and belief of the data-scientist about them. The observation of data leads to an update procedure, based on the Bayes' theorem, that allows such probability distribution of parameters to become more and more appropriate, each time exploiting the new available information. For both approaches, the maximum-likelihood is an optimal statistical principle and is widely exploited to choose parameters, in the frequentist approach, or to update parameters' probability distributions in the Bayesian one. Up-to-now the Bayesian inference has never explicitly influenced the Side-Channel Attack literature, nor we will use such a framework in this thesis. We leave this track opened for future works, briefly discussing its suitability for Side-Channel Attacks domain in Chapter 7.

2.2 Side-Channel Attacks

Side-Channel Attacks (SCA) belong to the cryptanalysis domain, since they aim to breach cryptographic security systems. Usually their goal is to retrieve a secret variable of a cryptographic algorithm, typically a secret key. They distinguish from classic mathematical cryptanalysis techniques by the fact that they are based on information gained from the physical implementation of a cryptosystem, rather than theoretical weaknesses in the algorithms.

2.2.1 Attacks general strategy

2.2.1.1 Grey-box and divide-and-conquer

Side-Channel Attacks go beyond the cryptographic complexity that assures resistance against classical cryptanalysis strategies. Indeed, no matter the size of the secret variables manipulated by the algorithm and the algebraic complexity of the encrypting/decrypting operations, an hardware implementation of any algorithm always handles variables of a bounded size, that depends on the hardware architecture. For example, in an 8-bit architecture an AES with 128-bit-sized key will be necessarily implemented as multiple partial computations over 8-bit blocks of data. In classical cryptanalysis, the typical attacker model faces to a black-box that performs the cryptographic algorithm: an attacker may make queries to the black-box, asking for ciphertexts of given plaintexts or viceversa, but has no information about any partial computations. On the contrary, a side-channel attacker is said to face to a grey-box model: he has a way to obtain noised information about partial computations. This allows him to perform a *divide-and-conquer* strategy: if his goal is to retrieve the full 128-bit AES key, he will smartly divide his problem into the recovery of small parts of such keys at time, called *key chunks*,[1] making the complexity of the attack significantly drop.

2.2.1.2 Sensitive Variable

The noised information an attacker accesses, comes in form of side-channels signals, *e.g.* instantaneous power consumption or electromagnetic irradiation, that he acquires during the execution of the algorithm. Such signals are collected into vectors called *traces* (or *acquisitions*). They will be denoted by $\vec{x}_i$ and considered as observations of a random real vector $\vec{X}$, where each coordinate corresponds to a time sample of the acquired signal. The goal of the side-channel analysis is to clear traces from noise, in such a way to determine with the highest possible precision the association between a trace (or a set of traces) and the value taken by a target *sensitive variable* Z during its (their) acquisition. A sensitive variable is a quantity handled during the processing that tells something about a secret of the implementation. Actually, it would be better to call it *sensitive target*, since it might not be variable. Some typical examples of sensitive variables include:

- $Z = K$ with K a secret subkey - this is the most direct choice for a sensitive target, nevertheless it is often not variable, since in some cases a device always manipulates the same key for a given embedded primitive. When the target is not variable we are performing a *simple attack* (see below);

[1] or *subkeys* when they coincide to a byte of key for the AES algorithm

- a cryptographic variable that depends on a sufficiently small subkey and a part of a known input variable E: $Z = f(K, E)$ - this is the most classical choice to perform a so-called *differential* or *advanced* SCA (see below);
- any function of a cryptographic variable (ex: $\mathrm{HW}(f(K, E))$), where $\mathrm{HW}(\cdot)$ represents the Hamming weight operation, *i.e.* the operation that counts the number of 1's in the binary string representing the entry. Sometimes, as for example we will see in Chapter 5 (see Sec. 5.4.3) it can be interesting not to target a variable but a non-injective function of a variable as the Hamming weight; when the identity function is applied we are in the previous case;
- an operation (ex: $Z \in \{\mathrm{square}, \mathrm{multiply}\}$)
- a register (ex: Z is the register used to store results of intermediate operations in a Montgomery ladder implementation of RSA)

In this thesis we will try as much as possible to abstract from the form of the sensitive variable, thinking of any entity Z that assumes values in a finite set $\mathcal{Z}$ and whose value permits an attacker to make inference on a secret of the implemented algorithm. Anyway, the choice of the sensitive variable is crucial for the successfulness of an attack. Interestingly, many notions and a wide and still-evolving taxonomy of SCAs' typologies is in facts related to the choice of the sensitive variables. Some of these notions are reported in next three paragraphs by way of example.

Vertical vs Horizontal SCAs. Attacks in which sensitive information is extracted from a single acquisition split into several parts are called *horizontal*. Horizontal attacks may apply when several internal variables depend on the same sensitive variable (see for instance [Bat+16]). Algebraic SCAs (see below) are horizontal attacks as well, in which inferences are not done over a single sensitive variable, but potentially over every cryptographic variable, followed by a deep analysis of the algebraic relations between these sensitive variables. Horizontal attacks differ from the so-called *vertical* attacks where information is obtained from different algorithm executions. The nice notion of rectangle attack has been introduced in [Bau+13] to refer to attacks that exploit both vertical and horizontal leakages.

Simple vs Advanced SCAs. A simple attack is an attack that only needs one trace to be applied, except for eventually profiling acquisitions (see profiling attacks later). Such a one-trace attack can be seen as a classification problem in machine learning language: an attacker guesses the value of the secret key from the observation of a single side-channel trace; in other words, setting Z equal to the definitive secret the attacker is looking for (*e.g.* the whole secret key), the attack consists in classifying one observation, *i.e.* assigning to the single attack trace the corresponding value of Z. In 2002 Mangard *et al.* [Man02] proposed for example a simple power analysis (SPA) strategy, improved by Clavier *et al.* in 2014 [CMW14], to retrieve the whole AES key observing leakages from a single execution of the AES key expansion. When Algebraic SCAs (see below) appeared in literature in 2009, their aim was to be a strategy to perform simple attacks as well.

Remark 2.1. Attacks for which many observations are acquired with fixed entry parameters and by consequence in which the observed leakage always corresponds to a fixed Z are still considered as simple attacks. The attacker may exploit the several acquisitions in mainly two ways: he computes their average before performing the

classification, aiming to reduce the noise influence, or he performs the classification of each acquisition (expecting each gives the same outcome) and then applies a function to the several outcomes (*e.g.* majority vote or computation of joint probabilities, if outcomes are probability distributions, see 2.11) to guess the right label. We observe that this approach with several observations allows the attacker to reduce the noise impact, while observing many times the same variable. The relation between the variable and the key chunk being fixed, he will not exploit algebraic relations to ameliorate his inference over the latter.

An advanced attack is a more powerful strategy: the attacker acquires several acquisitions making entry parameters vary, and by consequence observing leakages related to different values of Z. This time the variation of the observed sensitive variable is interpretable as a raising of the amount of caught information. The attacker exploits synergistically the information coming from each acquisition: he evaluates each key hypothesis, taking advantage of the algebraic relation between (known) entries, keys and sensitive variables, to find out the one that would better justify the leakages he observed. Classical Differential Power Attacks (DPA) [BCO04] or Correlation Power Attacks (CPA) [BCO04], as well as attacks based over Mutual Information Analysis (MIA) [Bat+11] are advanced attacks.

Algebraic Side-Channel Attacks. Algebraic SCAs (ASCA) were firstly proposed in 2009 [RS09; RSVC09]. Their aim was to combine profiling SCA strategies (see 2.2.3) to classical cryptanalysis techniques: in the latter seminal papers block ciphers implementations are attacked, and the authors added to plaintext and ciphertext knowledge, classically assumed in algebraic cryptanalysis, the access to the (exact) Hamming weights of several intermediate computations, obtained by side-channel observation. Once recovered as many partial information as possible, it is expressed in the form of an equation system, then converted to a set of clauses treatable by a SAT solver. In opposition to classic SCAs, ASCA approach does not exploit a divide-and-conquer strategy: the whole secret key is retrieved at once, exploiting algebraic relations between intermediate variables. This means, in the case of block ciphers, that the attacker observes leakages occurring during each round of the algorithm, and not only those related to the first rounds, were the cryptographic diffusion is limited. This optimal exploitation of the information, allows the ASCA strategy to provide efficient simple attacks, *i.e.* succeeding with a single attack trace.
This approach presents two main weaknesses. First, it does not tolerate errors, implying that it is weak to noise: a wrong side-channel information may put the right key out of the list of key candidates. This is why seminal ASCA papers proposed to equip the strategy with some techniques of detection of impossibilities and likelihood rating. Second, the use of SAT solvers asks to express relations between cipher variables at a bit level. Since general block ciphers are byte-oriented this produces very large and complex instances, challenging to construct and hard to debug.
Two works appeared in 2014 address these two weaknesses. In [VGS14] a *Soft Analytical Side Channel Attack* (SASCA) is proposed to make the ASCA strategy more tolerant to noise. The idea of such SASCAs is to replace the equation system representation of retrieved intermediate variables with a code, inspired by the low density parity check codes. This approach is still bit-oriented. Then the code is efficiently decoded by an algorithm known as *Belief Propagation*, whose inputs are not the exact values of the retrieved intermediate variables, but their probability distributions, provided by the profiling phase. The noise tolerance is provided by the utilisation of such probabilities instead of exact values.

In [OWW14] a constraint solver is proposed to replace the bit-oriented SAT solver. Such a new solver is designed for side channel cryptanalysis of byte-oriented ciphers, and works in a probabilistic way, as well as the SASCA approach, tracking the likelihoods of values in the secret key. The likelihoods of observed intermediate variables are provided by a template approach, as well as proposed in **[Oren:2013]** in 2013 under the name of Template-Algebraic SCA (TASCA). The main tool of the constrained solver is the *conflation operator* for reconciling multiple probability distributions for the same variable.

Since all these algebraic side-channel approaches are based over a preliminary profiling phase, they are all largely concerned by the dimensionality reduction issue, which is often not explicitly taken into account in the literature about this research axe.

2.2.1.3 Leakage Models

The underlying hypothesis of a SCA is that some information about a sensitive variable of the implemented algorithm leaks during its execution through some observable *side* channels. Such leakages are collected in the form of signal traces. Depending on the observed channel (*e.g. power consumption, electromagnetic irradiation, time, ...*), different properties might influence the form of the leakage, and should be taken into account for the construction of a leakage model.

If we allow a Side-Channel attacker to make use of a probing station to directly access the circuit wires and monitor the exact values of some intermediate values, this attacker will observe leakages following the so-called *probing model*. To define this model no further hypothesis is needed, for example no noise is taken into account. As explained in 1.2.1.2, SCAs typically refer to non-invasive attacks, so in Side-Channel literature the probing model has been introduced as a worst-case abstract model, and is mainly considered in order to provide formal security proofs for some kinds of countermeasures. More precisely the d-probing model [ISW03], in which an attacker can probe d different wires at a time, provides a good model to exhibit security proofs for dth-order masking schemes (cf. 2.3.2).
The most common passive leaking channel considered in literature is the power consumption. For such a physical quantity many efforts have been done to propose adherent leakage models. A detailed modelling for power consumption of CMOS circuits is proposed in the *DPA book* [MOP08]. After a description of the physical factors influencing the power consumption (divided into static and dynamic) of single logic cells, the authors propose to assume two different points of view to model and develop simulations of the power consumption: the designer point of view can bring to a quite accurate and detailed model, essentially based over his circuit transistor netlists. On the contrary an attacker would be satisfied by considering some easier model, often based over the *Hamming distance* (HD) or the *Hamming-Weight* (HW) of internal variables. Indeed these two functions well-fit the consumption behaviour of circuits registers and buses, which consume depending on how many bits set to 1 or 0 they store or transport (Hamming weight) or how many of them switch their value for 0 to 1 or vice-versa during computations (Hamming distance).
When an attacker has chosen its sensitive target Z and deals with concrete acquisitions, he does not need a complete power model, but only a way to model the relative differences between leakages for different values of Z, in order to distinguish traces related to different values of Z, or in other terms, closer to the machine learning

language, to classify traces depending on their associated value of Z. A statistical model is then sufficient to him. Thus for an attacker, the wider considered model in Side-Channel community is the one sometimes called *noisy leakage model*, firstly introduced in [Cha+99], that considers that the physical observable is a deterministic function of the sensitive variable, corrupted by some noise. Some efforts to better specify the form of the noise in such a model have been done in literature, often with the aim of providing security formal proofs for side-channel countermeasures. This is the case for the seminal publication [Cha+99], where noise is assumed following a Gaussian distribution, and is thus quantified by its standard deviation, and for the model generalization proposed in [PR13] where noise is quantified as a statistical distance, called *bias*, between the distribution of Z and the conditional distribution of Z given $\vec{X}$. In this thesis we do not need to consider a formal description of the noise, nor a precise concept to quantify it. Despite the fact that some of the proposed techniques present optimality features in presence of Gaussian hypothesis, we will not endorse the Gaussian model. To us, side-channel traces are informative, in the sense that the conditional probability densities of $\vec{X}$ given Z are different for a least two different values assumed by Z. We assume in particular that such difference is probability density is observable in their first-order moment, *i.e.* the expected value. The role of the attacker is to understand how to optimally take advantage of these statistical differences, minimizing the number of observations he needs to succeed, and maximizing the efficiency of the attack, in the sense formalized in next section.

2.2.2 Efficiency of the SCAs

In order to measure the efficiency of a side-channel attack, different security metrics have been proposed, the most exploited being the *success rate of order* o (SR_o) and the *guessing entropy* (GE). Referring to the formalization proposed by [SMY09], a key recovery side-channel attack outputs a vector of key candidates,[2] called *guessing vector* $\vec{g} = [\vec{g}[1], \ldots, \vec{g}[|\mathcal{K}|]]$, in which such candidates are sorted in decreasing order with respect to their likelihood after the attack phase. Being $k^\star$ the right candidate, its *rank* is given by:

$$\mathrm{Rank}(k^\star) = i \text{ such that } \vec{g}[i] = k^\star. \tag{2.6}$$

Then, the success rate of order o of an attack is given by the probability for the right key candidate to be ranked among the first o candidates:

$$\mathrm{SR}_o = \mathrm{Prob}[\mathrm{Rank}(k^\star) \leq o] \,. \tag{2.7}$$

The success rate of an attack is usually estimated empirically: the attack is repeated a large number of times, and the empirical SR_o is given by the ratio between the number of successes (attacks for which the right key is ranked among the first o ones) and the total number of attacks.

The guessing entropy [Mas94] is defined as the expected rank of the right key:

$$\mathrm{GE} = \mathbb{E}[\mathrm{Rank}(k^\star)] \,. \tag{2.8}$$

This is also generally estimated in an empirical way, by performing the attack many times independently, then computing the average of the obtained ranks.

[2] In this thesis we will always target a key chunk and we will use such metrics to evaluate the efficiency of an attack in recovering such key chunks. When a full-key recovery attack is run, some algorithms to merge key chunks' outcomes and obtain the full key enumeration and a complete key rank estimation are deployed. This domain is out the scope of this thesis.

2.2.3 Profiling Side-Channel Attacks

As anticipated in Sec. 1.3, when an open sample of the attacked device is available to make a prior characterisation of the leaking signals of a device, we talk about *profiling* attacks. When this is not the case, we talk about *non-profiling* attacks.
A profiling attack is divided into two distinct phases. The first one, called *profiling phase* or *characterisation* phase exploits so-called *profiling traces* to build a model of the leakages. Profiling traces are acquisitions taken under known values for the sensitive variable Z, so are couples $(\vec{x}_i, z_i)_{i=1,\dots,N_p}$ for which the correct association trace/sensitive variable is known. The second phase of a profiling attack is the proper *attack phase*, during which the attacker observes a new set of acquisitions, under unknown secret key, and takes advantage of the previous characterisation to infer over it. Throughout this thesis, and each time a profiling attack scenario is supposed, we will refer to elements of $\mathcal{Z}$ as *labels*, each one identifying a *class* of traces. We will say that acquired traces associated to a same value $z \in \mathcal{Z}$ *belong* to the same class, identified by the label z. We will say as well that such traces are *labelled* by the value z. By abuse we will also refer to the class z to denote the class of traces labelled by z. In such a context N_z will denote the number of profiling traces belonging to the class z.

As we will see in Chapter 3, in machine learning domain the analogous of profiling attacks context is studied under the name of *supervised machine learning*. In supervised machine learning, couples $(\vec{x}_i, z_i)_{i=1,\dots,N_p}$ are available and are called *training examples*. The profiling phase is referred to as *training* or *learning* and the attack phase is assimilable to the so-called *test phase*. The main difference between a machine learning test phase and a side-channel attack phase is that in the former one the examples are processed independently from each other, while in the latter the examples have something in common (typically a fixed secret key) and are used synergetically to guess it. If no example is available we talk about *unsupervised machine learning*, that we can consider analogous to the non-profiling SCAs branch.

2.2.3.1 Template Attack

Introduced in 2002 by Chari [CRR03], the so-called *Template Attack* (TA) is the most well-established strategy to run a profiling SCA. It can be performed in a simple or advanced way. The idea of the TA is based over the construction of a so-called *generative model*: in probability, statistics and machine learning " ...approaches that explicitly or implicitly model the distribution of inputs as well as outputs are known as generative models, because by sampling from them it is possible to generate synthetic data points in the input space." [Bis06]. In TA the attacker observes the couples $(\vec{x}_i, z_i)_{i=1,\dots,N_p}$ and exploits them to estimate the class-conditional densities

$$p_{\vec{X} \mid Z=z}(\vec{x}) \,, \tag{2.9}$$

eventually the prior densities $p_{\vec{X}}(\vec{x})$, $p_Z(z)$, and finally the *a-posteriori* density, by means of Bayes' theorem:

$$p_{Z \mid \vec{X}=\vec{x}}(z) = \frac{p_{\vec{X} \mid Z=z}(\vec{x}) p_Z(z)}{p_{\vec{X}}(\vec{x})} \,. \tag{2.10}$$

In the attack phase the attacker acquires new traces that he only can associate to the public parameter E, obtaining couples $(\vec{x}_i, e_i)_{i=1,\dots,N_a}$. Then he makes key hypothesis $k \in \mathcal{K}$ and, making the assumption that each acquisition is an independent

observation of $\vec{X}$, he associates to each hypothesis $k \in \mathcal{K}$ a score d_k given by the joint *a-posteriori* probability that follows, and that he computes exploiting estimates (2.10):

$$d_k = \prod_{i=1}^{N_a} p_{Z \mid \vec{X}=\vec{x}_i}(f(k, e_i)) \,. \tag{2.11}$$

Finally, his best key candidate $\hat{k}$ is the one maximizing such a joint probability

$$\hat{k} = \underset{k}{\operatorname{argmax}}\, d_k \,. \tag{2.12}$$

Remark 2.2. Since the marginal probability density $p_{\vec{X}}(\vec{x}_i)$ of (2.10) does not depend on key hypothesis, it is usually neglected. Moreover, in many cases the variable Z follows a uniform distribution, so its probability mass function $p_Z(z)$ appearing in (2.10) does not influence the ranking of key hypothesis. It is often neglected as well.

Remark 2.3. In the special case of a simple attack, *i.e.* $N_a = 1$, in which $Z = K$, the problem becomes a classical machine learning classification problem (as we will discuss over in Chapter 3): the attacker wants to classify the unique attack trace, *i.e.* assign to it a class label (the key). In such a case, the choice proposed by (2.12) is known as *Bayes (optimal) classifier*.[3] It is proven to be the optimal choice to reduce the misclassification error [Bis06].

This approach has the theoretical optimality that comes from the maximum likelihood criterion. The crucial point is the estimation of the class-conditional densities (2.9): the efficiency of the attack strongly depends on the quality of such estimates.

The Gaussian Hypothesis. A well-established choice to construct class-conditional densities estimations 2.9 is the one applied in Gaussian TA [CRR03]: it consists in making a class-conditional multivariate Gaussian distribution assumption

$$\vec{X} \mid Z = z \sim \mathcal{N}(\vec{\mu}_z, \Sigma_z) \,, \tag{2.13}$$

and exploits the profiling traces to estimate the parameters $\vec{\mu}_z$, *i.e.* the mean vector of the Gaussian distributions, and Σ_z, *i.e.* the covariance matrices.

Remark 2.4. This assumption is the same that is done for classification problems, bringing to the *Quadratic Discriminant Analysis* technique, which we will describe in Chapter 3.

Many options and choices influence the implementation of a TA: the suppression or not of the marginal densities in (2.10), the use of the unbiased estimator or the maximum likelihood estimator for the covariance matrices, the addition of an *homoscedasticity* assumption (assume that all class-covariance matrices are equal). This last assumption, proposed in 2014 in SCA literature [CK14b], allows exploiting all profiling traces to estimate a unique so-called *pooled* covariance matrix, instead of using traces belonging to each class to estimate each covariance matrix separately. The pooled estimation gains in accuracy.

[3] The term *optimal* distinguishes it from the so-called *Bayes naive classifier*, which introduces an independence assumption between data vector coordinates. The efficiency of a Bayes naive classifier has been analysed in SCA context in 2017 [PHG17].

Remark 2.5. The homoscedasticity assumption is the same that is done for classification problems, bringing to the *Linear Discriminant Analysis* technique, which we will introduce in Chapter 3 and more deeply analyse in Chapter 4.

Other choices that mainly influence the TA efficiency are those related to the PoI selection, or more generically to the dimensionality reduction issue.

2.2.4 Points of Interest and Dimensionality Reduction

Side channel traces are usually acquired by oscilloscopes with a very high sampling rate, which permits a powerful inspection of the component behaviour, but at the same time produces huge-dimensional data, consisting in thousands, or even millions of points. Nevertheless, often only a relatively small part of these time samples is informative, i.e. statistically depends, independently or jointly, on a sensitive target variable. These informative points are called *Points of Interest* (PoI). The dimensionality reduction of the traces is a fundamental pre-processing phase to get efficient and effective SCAs, not too expensive in terms of memory and time consumption. The problem of performing an opportune dimensionality reduction goes hand in hand with the research of PoIs: a convenient dimensionality reduction should enhance the contribution of such PoIs while reducing or nullifying the one provided by non-interesting points. The goal of researches in this context is to study and develop techniques to characterise PoIs and to apply convenient dimensionality reduction techniques, that allow reducing the size of the acquisitions while keeping the exploitable information held by data high enough to allow an SCA to succeed. Representing the side channel traces as column vectors $\mathbf{x}$ in $\mathbb{R}^D$, the compressing phase might be seen as the application of a function $\epsilon\colon \mathbb{R}^D \to \mathbb{R}^C$, with $C < D$, called *extractor* throughout this thesis. The first extractors proposed in SCA literature where actually some selectors of time samples, *i.e.* functions that operate a simple subsampling of the traces on the base of the computation of some sample-wise statistics $\tau(t)$, whose aim is to quantify a sort of signal strength. Several proposals exist for such a signal-strength estimate, among them the most deployed are the Difference of Means (DOM) [CRR03], or the analogous but better specified Sum of Differences (SOD) [RO05], the Sum of Squared Differences (SOSD) [GLRP06], the Signal-to-Noise Ratio (SNR) [MOP08; LPR13] and Sum of Squared t-differences SOST, corresponding to the t-test [GLRP06]. All these statistics are similar, and exploit the sample mean per class of the traces, given by

$$\overline{\vec{x}}^z = \hat{\mathbb{E}}[\vec{X} \mid Z = z] = \frac{1}{N_z} \sum_{i\colon z_i = z} \vec{x}_i \,. \tag{2.14}$$

A notable difference among them is that only the last two, SNR and SOST, take also the variances per class of the traces into account, given by

$$\hat{\boldsymbol{\sigma}}_z^2 = \hat{\mathrm{Var}}(\vec{X} \mid Z = z) = \frac{1}{N_z - 1} \sum_{i\colon z_i = z} (\vec{x}_i - \overline{\vec{x}}^z)^2 \,. \tag{2.15}$$

The Table 2.1 gives explicit formulas to compute such state-of-the-art sample-wise statistics. Once the chosen signal strength estimate τ is computed, it can be used as in a hypothesis test to reject the hypothesis that the sample mean values at time t are equal. The instants t in which such a hypothesis is rejected correspond to the PoIs, since the variation of the signals in such instants seems depend on the class belongingness. The construction of the subsampling ϵ is done on the base of such

TABLE 2.1: Statistics proposed as signal strength estimate to operate a selection of time samples.

Name of the estimate	Definition
SOD	$\tau(t) = \sum_{\substack{z_1,z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\overline{\vec{x}}^{z_1}(t) - \overline{\vec{x}}^{z_2}(t))$
SOSD	$\tau(t) = \sum_{\substack{z_1,z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\overline{\vec{x}}^{z_1}(t) - \overline{\vec{x}}^{z_2}(t))^2$
SOST (version [GLRP06])	$\tau(t) = \frac{\sum_{\substack{z_1,z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\overline{\vec{x}}^{z_1}(t) - \overline{\vec{x}}^{z_2}(t))^2}{\frac{\hat{\boldsymbol{\sigma}}^2_{z_1}}{N_{z_1}} + \frac{\hat{\boldsymbol{\sigma}}^2_{z_2}}{N_{z_2}}}$
SOST (version [BDP10])	$\tau(t) = \frac{\sum_{\substack{z_1,z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\overline{\vec{x}}^{z_1}(t) - \overline{\vec{x}}^{z_2}(t))^2}{\hat{\boldsymbol{\sigma}}^2_{z_1} + \hat{\boldsymbol{\sigma}}^2_{z_2}}$
SNR	$\tau(t) = \frac{\hat{\mathrm{Var}}(\overline{\vec{x}}^{Z}(t))}{\hat{\mathbb{E}}[\hat{\boldsymbol{\sigma}}^2_{Z}(t)]}$ (2.16)

a test, for example selecting all time samples for which $\tau(t)$ is higher than a certain threshold.

As anticipated in Sec. 1.3.2, in this thesis we did not go deeper in the study of such sample-wise PoI selection methods, exploring directly other dimensionality reduction approaches. Anyway, throughout the thesis, we will often refer to the SNR statistic, as a good indicator of sample-wise information.

2.3 Main Side-Channel Countermeasures

To counteract SCAs, strategies that aim at making leakages independent from the processed sensitive data have to be implemented. We can distinguish two broad groups of such countermeasures: those that aim at hiding the data and those that are designed to mask the data. The two approaches may even be combined.

2.3.1 Hiding

The main characteristic of a hiding countermeasure is that is does not change the intermediate data values that are processed in the cryptographic algorithm, but it only attempts in hiding its processing. Hiding is typically, but not only,[4] achieved in by randomising the power consumption. A random power consumption can be

[4]Strategies to attempting making power consumption constant, such as the use of dual-rail precharge logic cells, also belong to the hiding group of countermeasures [PM05].

obtained by randomly changing the time at which the targeted sensitive variable is processed. In this way the attacker acquires side-channel traces that are desynchronised or misaligned with respect to their interesting part. This temporal misalignment reduces the effectiveness of an attacker's statistical analysis. Possible ways for randomising the power consumption are the random insertion of dummy instructions [CK09; CK10] and the shuffling of the operations [VC+12], at a software level, or the randomization of the instruction stream by means of non deterministic processors [IPS02; MMS01], or the enhancement of a jittering effect over the clock via an asynchronous logic style at a hardware level [Moo+02; Moo+03]. Such methods may also be combined.

Applying realigning preprocessing techniques, such as integration [Man04; MOP08], pattern matching [Nag+07] or more sophisticated signal-processing techniques [WWB11], is the most common approach an attacker usually chooses to face up temporal misalignment. Defeating differently misalignment countermeasures is one of the main motivations that lead us to investigate Convolutional Neural Networks, as we will discuss in Chapter 6.

2.3.2 Masking

Masking countermeasures derive from the idea of applying secret-sharing methods to counteract side-channel attacks. Secret-sharing methods consist in strategies to distribute a secret message amongst a group of participants. Each participant receives a piece of information, called *share* and the original message can only be reconstructed if a sufficient number of participants collaborate, putting in common the knowledge of a sufficient number of shares. The idea of applying secret-sharing to counteract SCAs was first proposed by Chari *et al.* [Cha+99] and Goubin and Patarin [GP99]. In this case the sensitive variables of the cryptographic algorithm are considered as secret messages to distribute. Since 1999, several masking schemes have been proposed, attacked and ameliorated to protect various cryptographic algorithm, for example [Mes00a; AG01; ISW03; BGK04; Osw+05; SP06; RP10; Mor+11; Cor+13; Bil+14; DC+15; GR17; JS17]. When a masking scheme is properly implemented, it guarantees that every sensitive variable Z is randomly split into multiple shares $M_1, M_2, \ldots, M_d$ in such a way that a relation

$$Z = M_1 \star \cdots \star M_d \tag{2.17}$$

holds for a group operation $\star$ (*e.g.* the exclusive or for the most popular Boolean masking already proposed in the seminal papers [Cha+99; GP99]). The soundness of the masking countermeasure is implied by the fact that, in the noisy leakage model, the complexity of recovering information by SCA on a bit shared into several pieces grows exponentially with the number d of shares.[5] This fact was enlighten by Chari *et al.* in 1999 [Cha+99], then complemented by Prouff and Rivain in 2013 [PR13]. As a consequence of such an exponential complexity behaviour, the number d of shares plays the role of a security parameter for a masking scheme and the method is usually referred to as $(d-1)$th-order masking (since it involves $(d-1)$ random values, called *masks* and one value determined by the sensitive variable and the relation (2.17), which is sometimes referred to as *masked variable*). The shares are manipulated by distant parts of the circuit (especially if the countermeasure is implemented at a hardware level) or at different times (especially for software implementations of

[5] The exponential basis being proportional to the noise standard deviation.

the countermeasure). In this way an attacker, who is obliged to retrieve information coming from a sufficient number of shares to obtain some Z-dependent information, has to acquire many portions of signal to combine.

Attacks against the masking countermeasure are known as *Higher-Order* Side-Channel Attacks (HO-SCA), where the order usually refers to the number of independent information an attacker has to join to succeed. In general, to defeat a $(d-1)$th-order masking countermeasure, a dth-order attack has to be run. In the first literature about HO-SCA (for instance [Mes00b; WW04; JPS05; Osw+06] the order corresponded to the number of time samples of the signal the attacker combined to mount the attack, and the common idea was to compute some combining function of the d time samples and compare the outcome with some key-dependant predictions. Among the proposed combining functions, the centred product of the d points were showed to be the most efficient, at least under a Hamming Weight power consumption model [PRB09]. Actually, and for example when the countermeasure is implemented in hardware and shares are manipulated in parallel, sometimes the number of time samples to combine differs from the number d of shares [Pee+05; SPQ05]. So the definition of dth-order SCA has mutated in time (see for instance a different formalization in [PS08]). Today it is most-widely accepted to define a dth-order attack as an attack that looks for key-discriminant information in some dth-order statistical moment of the signal, while the number of time samples of the signals that participate to such a statistic defines the *multivariability* of the attack [Gie+10; Bat+11; Car+14]. For example a 2nd-order attack against a parallel implementation may be univariate if a single time sample is used to derive key-dependent information. In general for attacks against software implementations, a dth-order attack is usually d-variate. In such a case the research of interesting d-tuples of time samples still raises the complexity of the attacks. Even in the favourable case in which a profiling attack is allowed, two cases must be distinguished: the attacker has or not access to the masks values during profiling. In the former case the attacker can use the shares as target sensitive variables during the profiling phase, looking for PoIs for each one of them. Thus, the PoI research complexity grows only linearly with the number d of shares. In the latter case the attacker cannot infer independently over each share and classical tools for PoIs research are inefficient. This issue is the main motivation that leads us to consider solutions based over the Kernel Discriminant Analysis (KDA) tool, as we will discuss in Chapter 5.

Chapter 3

Introduction to Machine Learning

3.1 Basic Concepts of Machine Learning

Machine Learning (ML) is a field of computer science that groups a variety of methods whose aim is giving computers the ability of *learning* without being explicitly programmed. The more cited definition of *learning* has been provided by Mitchell in 1997 [TM97]: " A computer program is said to learn from experience E with respect to some task T and performance measure P, if its performance on T, as measured by P, improves with experience E."
Machine Learning groups a variety of methods essentially coming from applied statistics, and characterised by an increased emphasis on the use of computers to statistically estimate complicated functions. This allows Machine Learning to tackle tasks that would be too difficult to solve with fixed programs written and designed by human being. A Machine Learning algorithm is often said to "learn from data", in the sense that it is able to improve a computer program's performance at some task via a data observation experience.

3.1.1 The Task, the Performance and the Experience

The task. The task T is usually described in terms of how the machine learning system should process an *example* (or *data point*). An *example* is one datum $\vec{x} \in \mathbb{R}^D$, which is in turn a collection of *features* $\vec{x}[i]$, with $i = 1, \dots D$. In SCA context an example might be a side-channel trace, which is in turn a collection of time samples, that constitute its features. Some common ML tasks include these three examples:

- *Regression:* the computer is asked to predict a numerical value, given some input. The learning algorithm is thus asked to construct a function $F\colon \mathbb{R}^D \to \mathbb{R}$.

- *Classification:* the computer is asked to specify which class or category an input belongs to, being $\mathcal{Z}$ the set of the possible classes. The learning algorithm is thus asked to construct a function $F\colon \mathbb{R}^D \to \mathcal{Z}$. We remark that this task is similar to the regression one, except for the form of the output, since in general $\mathcal{Z}$ is a discrete finite set. A slightly variant solution to the classification task consists in constructing a function $F\colon \mathbb{R}^D \to \{0,1\}^{|\mathcal{Z}|}$, if elements of $\mathcal{Z}$ are expressed *via* the *one-hot encoding* (see 2.1). A variant of the classification task consists in finding a function F defining a probability distribution over classes.

- *Verification*: the computer is asked to state whether or not two given inputs are instances of a same class or category. For example, it may be asked to state if two hand-written signatures have been produced by the same person. The learning algorithm is thus asked to construct a function $F\colon \mathbb{R}^D \times \mathbb{R}^D \to \{0,1\}$.

A variant of such a task consists in finding a function F defining the probability of each pair of inputs being instance of a same class. This problem differs from the classification one essentially for the form of the input.

The functions constructed by a Machine Learning algorithm somehow describe and characterise the data form and distribution, thus are often referred to as *models*.

The performance measure. The performance measure P designs a quantification of the ability of the learning algorithm. Depending on the task T, a specific performance measure P can be considered. For tasks as classification or verification the more common measure is the *accuracy* of the model, *i.e.* the proportion of inputs for which the model produces the correct output. Equivalently, the *error rate* may be used as a performance measure P, *i.e.* the proportion of inputs for which the model produces an incorrect output. For the regression task the more common performance measure P is the so-called *Mean Squared Error* (MSE): it is computed by averaging over a finite set of examples, the squares of the the differences between the correct outputs and the ones predicted by the model.
One of the crucial challenges of Machine Learning is that we are usually interested in how well a learning algorithm performs in producing a model that fits new, unseen data. For this reason, the performances of a Machine Learning algorithm are usually evaluated over a so-called *test set*, *i.e.* a set of examples that have not been used for the learning (or *training*) phase.

The experience. The experience E describes the way data and information are accessed by the learning algorithm during learning. In this context we principally distinguish two families of learning algorithms:

- the *supervised* learning algorithms access to a dataset of examples, each associated in general to a *target* or *label*. The term supervised reflects the fact that the learning is somehow guided by an instructor that knows the right answer over the learning dataset;
- the *unsupervised* learning algorithms access to a dataset, without any associated target. They try to learn useful properties of the structure of the dataset.

In general, the nature of the task is strictly related to the kind of experience the learner is allowed; for example the classification or regression tasks are considered as supervised tasks, while examples of unsupervised tasks include *clustering* and *data representation* or *dimensionality reduction*. For example, the Principal Component Analysis, that will be discussed in Chapter 4 in the context of SCA, is a dimensionality reduction algorithm that might be seen as an unsupervised algorithm that learns a representation of data. We will see in Chapter 4 that for SCA context a supervised version of the PCA has been proposed as well.

3.1.2 Example of Linear Regression

The regression task is not of high interest for the rest of this thesis, but is the most direct example to keep in mind to understand some basic Machine Learning concepts, such as the underfitting and the overfitting (see 3.1.4). Let us introduce a linear regression model to tackle the regression task: we want to construct a linear function $F\colon \mathbb{R}^D \to \mathbb{R}$, that takes an input $\vec{x}$ and outputs $\hat{y} = \vec{w}^\intercal \vec{x}$, where $\vec{w} \in \mathbb{R}^D$ is a vector of *parameters* that have to be learned by a learning algorithm in order to well describe

some data.[1] Let $\mathcal{D}_{\cdot} = (\vec{\mathcal{X}}_{\cdot}, \mathcal{Y}_{\cdot})$ denote a dataset, where $\cdot$ can stand for train or test depending on the role of the dataset in the experience, and let $|\mathcal{D}|$ denote the size of the dataset. Let us store the examples contained in $\vec{\mathcal{X}}_{\cdot}$ into a matrix $\mathbf{M}_{\cdot} \in \mathbb{R}^{D \times |\mathcal{D}|}$ and the targets contained in $\mathcal{Y}_{\cdot}$ into a targets vector $\vec{y}_{\cdot} \in \mathbb{R}^{|\mathcal{D}|}$. Let a learned model predict targets y_i by outputting $\hat{y}_i = \vec{w}^{\intercal}\vec{x}_i$ and let them be collected in turn into a predicted targets vector $\hat{\vec{y}}_{\cdot}$. The MSE is given by

$$\text{MSE}_{\cdot} = \frac{1}{|\mathcal{D}|} \left\| \hat{\vec{y}}_{\cdot} - \vec{y}_{\cdot} \right\|_2^2 . \tag{3.1}$$

The performance measure for the learning algorithm is MSE_{test}, meaning that the goal for the learning algorithm is to find a parameter vector $\vec{w}$ which minimises MSE_{test}. Nevertheless, such an objective cannot be directly imposed, because the learning algorithm only experiences over the training set, and not over the test set. An intuitive way to act, that can be proven to be the maximum likelihood solution to the problem, is to minimise $\text{MSE}_{\text{train}}$ instead of MSE_{test}. This minimization can be obtained by solving an easy optimization problem. When a learning algorithm behaves as an optimization algorithm that minimises a given function, such a function is called *cost function*, or *loss function* or *objective function*. The solution to such an optimization problem can be given in closed form, by means of the pseudo-inverse matrix $\mathbf{M}^{+}$ of $\mathbf{M}_{\text{train}}$, as follows:

$$\mathbf{M}^{+} = (\mathbf{M}_{\text{train}}\mathbf{M}_{\text{train}}^{\intercal})^{-1}\mathbf{M}_{\text{train}} \tag{3.2}$$

$$\vec{w} = \mathbf{M}^{+}\vec{y}_{\text{train}}. \tag{3.3}$$

3.1.3 Example of Linear Model for Classification

In this thesis we point out a strict relationship between the profiling SCAs and the classification task in Machine Learning context. For this reason we introduce here a very brief overview of how classically such a task is tackled, by means of linear models.
Classifying means assigning a label $z \in \mathcal{Z}$ to an example $\vec{x} \in \mathbb{R}^D$, or equivalently divide the input space $\mathbb{R}^D$ in *decision regions*, whose boundaries are referred to as *decision boundaries*. Making use of a linear model signifies exploiting some hyperplanes as decision boundaries. Datasets whose classes can be separated exactly by linear decision boundaries are said to be *linearly separable*. Following the discussion kept by Bishop in [Bis06], two different approaches to tackle the classification task should be distinguished: the direct research for a discriminant function F that assigns to an example a label, or the prior construction of a probabilistic model. This second approach might in turn be distinguished into two options, depending on whether a generative model (see Sec. 2.2.3.1), or a discriminative model is constructed (*i.e.* only conditional probability densities of outputs given the inputs are modelled). For this example we consider a probabilistic approach, constructing a generative model. This example will allow on one hand to introduce some interesting functions, such as the *logistic sigmoid* and the *softmax*, that will play a role in the construction of neural networks (see Chapter 6). On the other hand, the example justifies the large exploitation of linear discriminant functions, even in contexts where the goal of the learning algorithm to make decisions directly, dispensing with any

[1] An affine model may be considered as well by adding a *bias*, leading to $\hat{y} = \vec{w}^{\intercal}\vec{x} + w_0$. This model is equivalently obtained by adding an additional component to $\vec{x}$, always set to 1 and by writing back $\hat{y} = \vec{w}^{\intercal}\vec{x}$ with $\vec{w} \in \mathbb{R}^{D+1}$.

probabilistic interpretation. Indeed, linear models come out naturally when adding some assumptions on the data distributions, as those that will be introduced below.

Constructing a generative probabilistic model implies modelling the class-conditional probabilities $p_{\vec{X}}(\vec{x} \mid Z = z^j)$ for $j \in \{1, \dots, |\mathcal{Z}|\}$ as well as the class priors $p_Z(z^j)$ and $p_{\vec{X}}(\vec{x})$. Let us first consider a 2-class context, *i.e.* $\mathcal{Z} = \{z^1, z^2\}$. Then the posterior probability for the class z^1 is the following:

$$\text{Prob}(z^1 \mid \vec{x}) = \frac{\text{Prob}(\vec{x} \mid z^1)\text{Prob}(z^1)}{\text{Prob}(\vec{x})} = \tag{3.4}$$

$$= \frac{\text{Prob}(\vec{x} \mid z^1)\text{Prob}(z^1)}{\text{Prob}(\vec{x} \mid z^1)\text{Prob}(z^1) + \text{Prob}(\vec{x} \mid z^2)\text{Prob}(z^2)} \,. \tag{3.5}$$

To compare the two classes, we can evaluate their *log-likelihood ratio* defined as:

$$a = \log\left[\frac{\text{Prob}(z^1 \mid \vec{x})}{\text{Prob}(z^2 \mid \vec{x})}\right] = \log\left[\frac{\text{Prob}(\vec{x} \mid z^1)\text{Prob}(z^1)}{\text{Prob}(\vec{x} \mid z^2)\text{Prob}(z^2)}\right] . \tag{3.6}$$

Then we might assign the label the class z^1 to $\vec{x}$ if and only if $a > 0$, which corresponds to take as decision boundary the surface defined by $\text{Prob}(\vec{x} \mid z^1)\text{Prob}(z^1) = \text{Prob}(\vec{x} \mid z^2)\text{Prob}(z^2)$. We remark that Eq. (3.4) rewrites as:

$$\text{Prob}(z^1 \mid \vec{x}) = \frac{1}{1 + e^{-a}} = \sigma(a) \,, \tag{3.7}$$

where the function σ is the so-called *logistic sigmoid*. This remark translates in the multi-class case, *i.e.* $|\mathcal{Z}| > 2$, in the following way: the posterior probability for each class z^j is given by

$$\text{Prob}(z^j \mid \vec{x}) = \frac{\text{Prob}(\vec{x} \mid z^j)\text{Prob}(z^j)}{\text{Prob}(\vec{x})} = \frac{\text{Prob}(\vec{x} \mid z^j)\text{Prob}(z^j)}{\sum_{k=1}^{|\mathcal{Z}|} \text{Prob}(\vec{x} \mid z^k)\text{Prob}(z^k)} = s(\mathbf{a})[j] \,, \tag{3.8}$$

where $\mathbf{a}$ is a $|\mathcal{Z}|$-dimensional vector, whose entries are given by

$$\mathbf{a}[j] = \log\left[\text{Prob}(\vec{x} \mid z^j)\text{Prob}(z^j)\right] \,, \tag{3.9}$$

and s is the so-called *softmax* function, or *normalised exponential*, that is defined, entry-wise by:

$$s(\mathbf{a})[k] = \frac{e^{\mathbf{a}[k]}}{\sum_{j=1}^{|\mathcal{Z}|} e^{\mathbf{a}[j]}} \,. \tag{3.10}$$

Let us now introduce two assumptions about the class-conditional densities:

(i) we will suppose they follow a Gaussian distribution with parameters μ_j, Σ_j,

(ii) and all class-conditional densities share the same covariance matrix $\Sigma_j = \Sigma$,

so that

$$p_(\vec{x} \mid Z = z^j) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\vec{x}-\mu_j)^\intercal \Sigma^{-1}(\vec{x}-\mu_j)} \,. \tag{3.11}$$

Under these assumptions, and considering probability densities and masses instead of probability values[2] Eq. (3.6) rewrites as:

$$a = \log\left[\frac{p_Z(z^1)}{p_Z(z^2)}\right] - \frac{1}{2}\mu_1^\intercal\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^\intercal\Sigma^{-1}\mu_2 - \vec{x}^\intercal\Sigma^{-1}(\mu_2 - \mu_1) = \vec{w}^\intercal\vec{x} + w_0, \quad (3.12)$$

where we set

$$\begin{aligned}\vec{w} &= \Sigma^{-1}(\mu_1 - \mu_2)\\ w_0 &= \log\left[\frac{p_Z(z^1)}{p_Z(z^2)}\right] - \frac{1}{2}\mu_1^\intercal\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^\intercal\Sigma^{-1}\mu_2.\end{aligned}$$

The quadratic terms in $\vec{x}$, that appears in the exponent of the Gaussian density, have cancelled thanks to the common variance assumption (ii), thus we obtain that the decision boundary for the 2-class problem, given by $a = 0$ is a $(D-1)$-hyperplane of the input space.[3] This way of choosing linear boundaries is known under the name of *Linear Discriminant Analysis*. Another way to view the same linear classification model is in terms of dimensionality reduction: intuitively, in the 2-class case[4] one can see the term $\vec{w}^\intercal\vec{x}$ in (3.12) as a projection of the input $\vec{x}$ onto a one-dimensional subspace of $\mathbb{R}^D$ which is orthogonal to the decision boundary mentioned above. Then, the classification of the obtained dimensionality-reduced examples is done by the means of a real-valued threshold (that would correspond to w_0, in the optimal case). It can be shown that the dimensionality reduction obtained by the Fisher criterion that we will deploy in Chapter 4, to which we will refer to as LDA dimensionality reduction by a widely accepted abuse, is equivalent to the dimensionality reduction obtained in this example, under both assumptions (i) and (ii).
Relaxing the assumption (ii) and allowing each class-conditional density $p(\vec{x} \mid z^j)$ to have its own covariance matrix Σ_j, then the cancellations seen above will no longer occur, and the discriminant a turns out to be a quadratic function of $\vec{x}$. This gives rise to the so-called *Quadratic Discriminant Analysis*, that we already mentioned in Chapter 2 for its analogy with Template Attacks.

Assumptions (i) and (ii) also lead to the following expression for the posterior probability for z^1, directly implied by (3.7):

$$\text{Prob}(z^1 \mid \vec{x}) = \sigma(\vec{w}^\intercal\vec{x} + w_0)\,. \quad (3.13)$$

Thus, such a posterior probability is given by the sigmoid acting to a linear function of $\vec{x}$. Similarly, for the multi-class case, the posterior probability of class z^j is given by the j-th entry of the softmax transformation of a linear function of $\vec{x}$. This kind of *generalised linear model* can be thus used in a probabilistic discriminant approach, where the posterior conditional probabilities are directly modelled from data without passing through the estimations of class-conditional densities and priors. Such a discriminative approach is the one that will be adopted in Chapter 6 when considering models constructed by neural networks.

[2] A formal justification the validity of formula above for continuous random variables is out of the scope of this section.
[3] An analogous result can be obtained in the multi-class problem.
[4] again extensible to the multi-class case

3.1.4 Underfitting, Overfitting, Capacity, and Regularization

Underfitting and Overfitting. As already said, the main challenge of ML is that the learning algorithms are in general allowed to experience over training data, but the models they output are asked to fit some unseen test data. Observing the training data, an ML algorithm sets the model parameters in order to raise the performances over the training set, or equivalently to minimise the so-called *training error*. Nevertheless, at the end of the learning process, the model performance is evaluated over the test set, by measuring the so-called *test error*. Thus, two factors determine how well a ML algorithm acts: its ability to reduce the training error, and its ability to reduce the gap between the training and the test error. When the former ability is not satisfactory we assist to the *underfitting* phenomenon: the model is not able to obtain a low training error, or the ML algorithm is not able to determine model parameters that make training error to be low. On the other hand, if the latter ability is not satisfactory we assist to the *overfitting* phenomenon: the gap between the training and the test error, called *generalization gap*, is too large.

Capacity. The property of a model that controls its underfitting or overfitting behaviour is the *capacity*. Roughly speaking the capacity of a model quantifies the complexity of the functions it can represent: a model with higher capacity can be parametrised in such a way to represent a higher complex function. For example a linear regression model is able to represent all linear functions. To raise its capacity, quadratic, cubic or general polynomial terms might be included, passing from a linear regression model to a *polynomial regression* one. It allows the model to represent respectively quadratic, cubic or polynomial functions as well.[5]

The polynomial regression provides a striking example to understand the underfitting and overfitting phenomena. Consider a problem in which the examples $(x_i, y_i)_{i=1,\dots,N}$ lies in $\mathbb{R} \times \mathbb{R}$ and the true underlying function is quadratic, perturbed by a small noise. Let the training set contain 4 data points, *i.e.* $N = 4$. Figure 3.1 shows the results of a linear, quadratic and cubic regression in such a case: in the figure, red circles represents the 4 training points, the blue line gives the learned model and the green points are test example. Above the plots the evaluation of the MSE over the training and test sets is given. We can observe that the linear predictor is underfitting, since the line passes quite far from both training and test points and its training error is quite high. On the contrary, the cubic predictor is overfitting: it perfectly fits the 4 training points (it is the Lagrange polynomial interpolating such 4 points) but shows a huge error in predicting new examples. The quadratic regression is obviously in this case the model exhibiting the optimal capacity to solve such a problem.

A very rough way to have an intuition about the capacity of a model is counting the number of its parameters: the capacity in general grows with the number of parameters. Some formal ways to quantify the capacity of a model have been provided in ML literature. The most well-known is the *Vapnik-Chervonenkis dimension*: it measures the capacity of a classifier as the cardinality of the largest set of

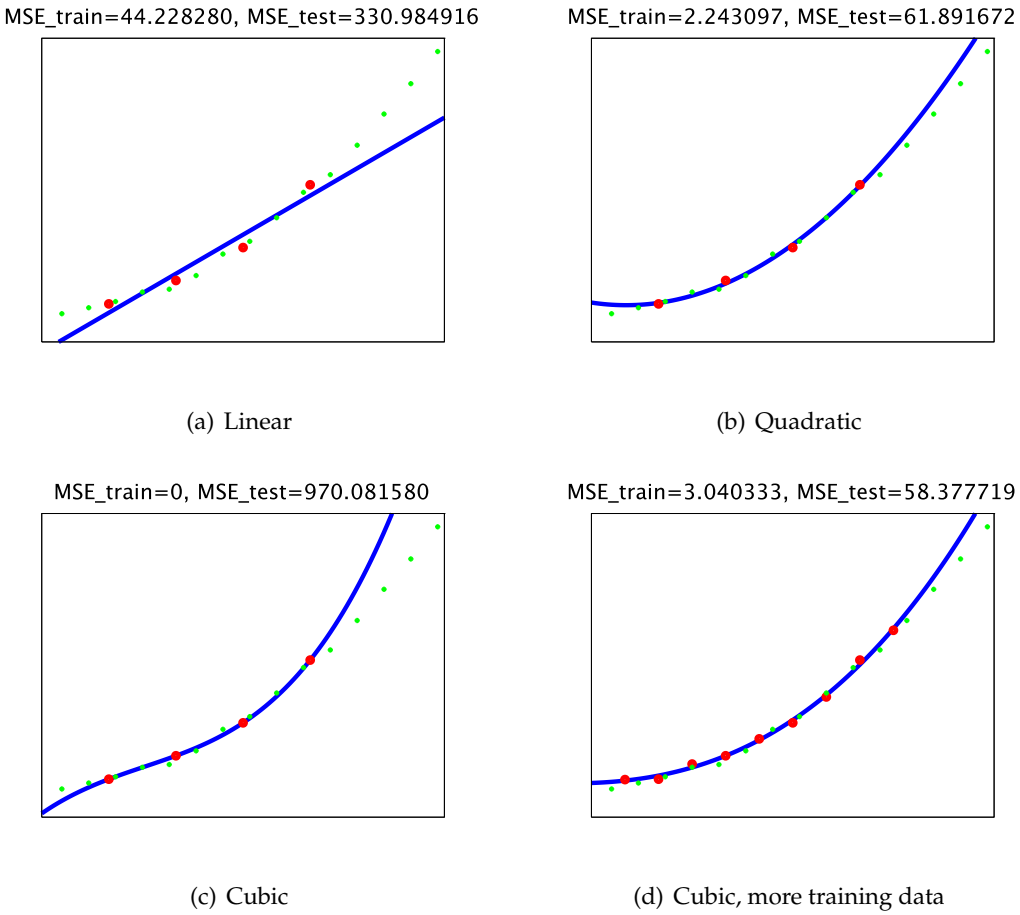
[5] Another common way to enlarge the capacity of a linear regression model $y = \vec{w}^\intercal \vec{x}$, consists in choosing some *basis functions* $\varphi_1, \varphi_2, \dots, \varphi_B$ and replace $\vec{x}$ with the values $\varphi_1(\vec{x}), \varphi_2(\vec{x}), \dots, \varphi_B(\vec{x})$. The form of the basis functions will determine the capacity of the model. Basis function regression includes the linear and the polynomial case.

points the model can classify without errors, for any possible assignment of labels. In practice, quantifying the capacity of a model, especially for complex models as those constructed by neural networks, is very hard and discouraged. On the other hand, these kinds of quantifications have enabled statistical learning theory to formalise and prove some important intuitions, for example the fact that the generalization gap is upper-bounded by a quantity that grows with the model capacity and that shrinks as the number of training examples increases. In Fig. 3.1(d) we observe how the cubic model used for regression on quadratic distributed data ameliorates its performances and reduces the generalization gap despite its excessive capacity, when trained with more examples. This observation basically justifies on one hand the attitude adopted in the branch of ML called *Deep Learning*, and basically based over multi-layer neural networks, consisting in considering very complex models, having confidence in the big size of the typically considered training sets. On the other hand it justifies the interest of *Data Augmentation* (DA) techniques to respond to an eventual lack of data. Some DA techniques will be proposed in Chapter 6 for the SCA context.

Regularization. In a real-case problem, the optimal capacity necessary to learn from given data is unknown. In such a case, trying to fit data with a too low capacity model assures the defeat, thus it is always more interesting to oversize the capacity of the learning model. Choosing an oversized model, we risk to incur in overfitting. The so-called *regularization* techniques respond to such a risk, as a widely adopted alternative to DA: in general they consist in adding constraints to the learning algorithm in order to guide it in choosing a model among a wide set of eventually fitting models. Going back to the polynomial regression example, one can try to fit data with a cubic polynomial (thus oversizing the model capacity) and induce the optimiser algorithm to choose the smallest-degree polynomial fitting data via a regularization. This can be obtained adding a penalty that depends on the polynomial degree to the cost function. Applying regularization may make the algorithm be less accurate in learning training data, but more likely to correctly operate on new examples.

3.1.5 Hyper-Parameters and Validation

The *hyper-parameters* of a model are all the parameters that are priorly set and that are not learned by the learning algorithm. They define the general form of the model. In the polynomial regression example the model had a single hyper-parameter: the degree of the polynomial. It is evident from the example that such a parameter is somehow forced not to be optimised by the means of the learning algorithm: trying to reduce the training MSE, the algorithm would choose a sufficient high degree to interpolate all training points (typically $N - 1$ if N is the number of training examples). This would cause overfitting, as shown in Fig. 3.1(c). In general among all parameters of a model, the hyper-parameters are chosen as those that can not be learned from data because it would cause overfitting, as in the example, or because they are too difficult to optimise.
A way to choose a setting for hyper-parameters consists in perform a *validation* phase. To do so the training set is split into two disjoint sets, one still called *training set* and the other one called *validation set*. We can say that as the training set is used to learn the parameters, the validation set is used to somehow learn the hyper-parameters. Indeed during or after the training over the training set, the validation set is used to compute a sort of estimation of the test error, which quantifies the



(a) Linear

(b) Quadratic

(c) Cubic

(d) Cubic, more training data

FIGURE 3.1: Examples of underfitting and overfitting over a regression problem. Linear (a), quadratic (b) and cubic regression for a truly noised quadratic problem. Red circles are the training examples, green points are the test ones, the blue line represents the learned solution. Linear (a) regression underfits data, cubic (c) regression overfits data. (d) Cubic regression for a noised quadratic problem and more training examples. The cubic model trained over more data is better adapted to the truly quadratic data, and overfitting is attenuated.

generalization ability of the model. In practice the performances of the (partially) trained model are evaluated over the validation set computing a validation error and hyper-parameters are updated accordingly, in order to reduce the generalization gap of the model. Once the model has been validated, *i.e.* the hyper-parameters are definitely set, the real test error is evaluated over the test set. Usually the validation error is an underestimation of the test error, since hyper-parameters have been set to reduce it.
The validation process just described may strongly depend on the way the training set have been split to create the validation one. In order to avoid to validate a model in a strongly data-dependent way, a slightly different process is encouraged in machine learning community, named the *cross-validation*, which we describe in Appendix A.

3.1.6 No Free Lunch Theorem

A so-called *No Free Lunch Theorem* has been formulated for optimization and machine learning algorithms around 1997 [WM97]. It states that any learning algorithm has the same test error if averaged over all possible distributions of data. This means that there cannot exist a universal best machine learning algorithm: any of them performs in the same way, when performances are averaged over all possible tasks. Thus, making research over some kind of data, for example SCA traces, means trying to understand what kinds of machine learning algorithms perform well over such particular kind of data and point out the eventual interesting hyper-parameters of machine learning models that are responsible of the main performance variations.

3.2 Overview of Machine Learning in Side-Channel Context

In 1991 Rivest pointed out for the first time a strong link between the fields of Machine Learning and Cryptanalysis [Riv91]. Starting from observing that the goal of cryptanalysis is identifying an unknown encryption function, indexed by a secret key, and that a classic problem in ML consists as well in learning an unknown function, he drew a strong correspondence between terminology and concepts of the two fields.

Machine Learning algorithms started to be investigated in Side-Channel Attacks context in 2011 [Hos+11]. In this paper the authors formulated for the first time an attack in terms of classification problem and proposed the Support Vector Machine (SVM) [CV95; WW98] as technique to solve it. They also equipped the SVM with a kernel function to allow it to succeed even in case data would not be linearly separable. Such an approach is similar to the one we will describe in Chapter 5, to obtain Kernel Discriminant Analysis dimensionality reduction technique from the Linear Discriminant Analysis. Further works analysed the use of SVM in SCA context, proposing concrete attack scenarios [HZ12; BLR13]. The technique of Random Forest [Lio+14] drew attention of the SCA community as well. As the SVM, it has been used as a classifier and has been evaluated in different works [LBM15; Ler+15; LBM14]. As in recent years the privileged tools to tackle classification problem in Machine Learning area are the Neural Networks, whose multi-layer configuration has given name to the so-called *Deep Learning* domain, such tools have as well been analysed in SCA context. Networks in the form of Multi-Layer Perceptrons (MLP) have been proposed as classifiers for side-channel traces in a series of

works [MHM13; MZ13; MMT15; MDM16], while Convolutional Neural Network was firstly introduced in [MPP16]. A part of this thesis contributions consists in the application of the convolutional paradigm as a way to defeat misalignment countermeasures in side-channel attacks (see Chapter 6).

Part II

Contributions

Chapter 4

Linear Dimensionality Reduction

We explore the solutions for dimensionality reduction of side-channel traces exploiting linear combinations of time samples. In this chapter we revisit the studies and results published as proceedings for CARDIS 2015 international conference [CDP16a]. The paper tackled some open questions regarding the two mainly explored linear dimensionality techniques, *i.e.* Principal Component Analysis and Linear Discriminant Analysis, proposed some solutions to such open questions, and performed a comprehensive comparison of the precedent and proposed methods in real cases, allowing verifying the soundness of the latter ones.

4.1 Introduction

Linear dimensionality reduction methods produce a low-dimensional linear mapping of the original high-dimensional data that preserves some feature of interest in the data. An abundance of methods has been developed throughout statistics, machine learning, and applied fields for over a century, and these methods have become indispensable tools for analysing high dimensional, noisy data, such as side-channel traces. Accordingly, linear dimensionality reduction can be used for visualizing or exploring structure in data, denoising or compressing data, extracting meaningful feature spaces, and more. A very complete survey about this great variety of linear dimensionality reduction technique has been published in 2015 by Cunningham and Ghahramani [CG15]. They proposed a generalized optimization framework for all linear dimensionality techniques, survey a dozen different techniques and mention some important extensions such as kernel mappings.

Among the surveyed methods in [CG15] we find the two mainly considered in SCA literature: the Principal Component Analysis (PCA) and the Linear Discriminant Analysis (LDA). The PCA has been applied both in an *unsupervised* way (*i.e.* non-profiling attacks) [BHW12; Kar+09], and in a *supervised* way (*i.e.* profiling attacks) [Arc+06; CK14a; CK14b; EPW10; SA08]. As already remarked in [EPW10] and not surprisingly, the complete knowledge assumed in the supervised approach hugely raises performances. The main competitor of PCA in the profiling attacks context is the LDA, that thanks to its classification-oriented flavour (see Sec. 3.1.3), is known to be more meaningful and informative [Bru+ a; SA08] than the PCA method for side channels. Nevertheless, the LDA is often set aside because of its practical constraints; it is subject to the so-called *Small Sample Size problem (SSS)*, i.e. it requires a number of observations (traces) which must be higher than the dimension (size) D of them. In some contexts it might be an excessive requirement, which may become unacceptable in many practical situations where the amount of observations is very

limited and the traces size is huge.

In 2014 Durvaux et al. proposed the use of another technique for linear dimensionality reduction in SCA context [Dur+15], the so-called Projection Pursuits (PPs), firstly introduced in 1974 by Friedman and Tukey [FT74]. This method essentially works by randomly picking time samples, randomly setting the projecting coefficients, and tracking the improvement (or the worsening) of the projection when modifying them with small random perturbations. The main drawback of the PPs pointed out by the authors of [Dur+15] for the SCA context is their heuristic nature, since the convergence of the method is not guaranteed and its complexity is context-dependent. The main advantage is the fact that PPs can deal with any objective function, which may be adjusted to fit the problem of higher-order SCA. Thus this technique appears advantageous in higher-order context, where it is used as a PoI selection tool. Its version for the first-order attacks, which produces a linear dimensionality reduction, is less interesting than the non-heuristic PCA and LDA. For this reason we will left PPs technique apart in this chapter, and describe their higher-order version in Chapter 5.

In SCA literature, one of the open issues for PCA application concerns the choice of the principal components that must be kept after the dimension reduction: as already remarked by Specht et al. [Spe+15], some papers declare that the leading components are those that contain almost all the useful information [Arc+06; CK14b], while others propose to discard the leading components [BHW12]. In a specific attack context, Specht et al. compares the results obtained by choosing different subsets of consecutive components, starting from some empirically chosen index. They conclude that for their data the optimal result is obtained by selecting a single component, the fourth one, but they give no formal argumentation about this choice. Such a result is obviously very case-specific. Moreover, the possibility of keeping non-consecutive components is not considered in their analysis.

In Sec. 4.2 the classical PCA technique is described, then the previous applications of PCA in SCA context are recalled, highlighting the difference between its unsupervised and supervised declination. Finally our contribution to the choice of components open issue is described: such a solution is based on the Explained Local Variance (ELV) notion, that we will define and argument in the same section. The reasoning behind the ELV selection methodology is essentially based on the observation that, for secure implementations, the leaking information, if existing, is spread over a few time samples of each trace. This observation has already been met by Mavroeidis et al. in [Mav+12], where the authors also proposed a components selection method. As we will see in Sec. 4.2.4, the main difference between their proposal and ours is that we do not discard the information given by the eigenvalues associated to the PCA components, but we synergistically exploit such information and the observation met. We will argue about the generality and the soundness of this methodology and show that it can raise the PCA performances, making them close to those of the LDA, even in the supervised context. This makes PCA an interesting alternative to LDA in those cases where the LDA is inapplicable. The ELV selection tool has been tested in a successive experimental work [CK18]. Unfortunately, the authors of such a work could not observe an improvement (nor a worsening) using such a selector, because in their specific case it returned the selection of the first components by eigenvalues. The LDA technique will be described in Sec. 4.3, together with the description of the SSS problem and some solutions coming from the Pattern and Face Recognition communities [BHK97; Che+00; Hua+02; YY01]. Through

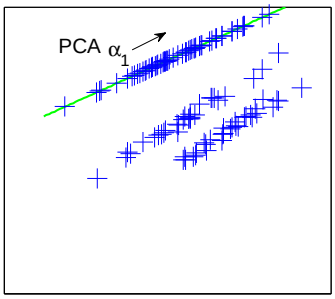


FIGURE 4.1: PCA: some 2-dimensional data (blue crosses) projected into their 1-dimensional principal subspace (represented by the green line).

some experiments depicted in Sec. 4.4 we will conclude about the effectiveness of the PCA-ELV solution. Finally, in Sec. 4.5 we will experimentally argue about the weakness of all these techniques to data misalignment.

4.2 Principal Component Analysis

4.2.1 Principles and algorithm description

The Principal Component Analysis (PCA) is a technique for data dimensionality reduction. The PCA algorithm can be deduced from two different points of view, a statistical one and a geometrical one. In the former, PCA aims to project orthogonally the data onto a lower-dimensional linear space, the so-called *principal subspace*, such that the variance of the projected data is maximized. In the latter, PCA aims to project data onto a lower-dimensional linear space in such a way that the average projection cost, defined as the mean square distance between the data and their projections, is minimized. In the following it is shown how the PCA algorithm is deduced by the statistical definition. The reader interested by the equivalence between the two approaches can refer to [Bis06, Ch. 12]. An example of 2-dimensional data projected over their 1-dimensional principal subspace is depicted in Fig. 4.1.

Let $(\vec{x})_{i=1..N}$ be a set of D-dimensional measurements (or observations, or data), i.e. realizations of a D-dimensional zero-mean random vector $\vec{X}$, and collect them as columns of an $D \times N$ matrix $\mathbf{M}$, so that the empirical covariance matrix of $\vec{X}$ can be computed as

$$\mathbf{S} = \frac{1}{N}\mathbf{M}\mathbf{M}^{\mathsf{T}} . \tag{4.1}$$

Let us assume for the moment that we have priorly fixed the dimension $C < D$ of the principal subspace we are looking for.

Compute the First Principal Component Suppose in a first time that $C = 1$, i.e. that we want to represent our data by a unique variable $Y_1 = \vec{\alpha}_1\vec{X}$, i.e. projecting data over a single $1 \times D$ vector $\vec{\alpha}_1$, in such a way the the variance of the obtained data is maximal. The vector $\vec{\alpha}_1$ that provides such a linear combination is called *first principal component*. To avoid misunderstanding we will call *j-th principal component* (PC) the projecting vector $\vec{\alpha}_j$, while we will refer to the variable $Y_j = \vec{\alpha}_j\vec{X}$ as *j-th Principal Variable (PV)*. Realizations of the PVs are given by the measured data

projected over the j-th PC, for example we can collect, in a (row) vector $\vec{y}_1 = \vec{\alpha}_1\mathbf{M}$, N realizations of Y_1:

$$y_1[i] = \vec{\alpha}_1\vec{x} \text{ for } i = 1, \ldots, N \,. \tag{4.2}$$

The mean of these realizations will be zero as they are linear combinations of zero-mean variables, and the variance turns to be expressible as

$$\frac{1}{N}\vec{y}_1\vec{y}_1^\intercal = \frac{1}{N}\vec{\alpha}_1\mathbf{M}\mathbf{M}^\intercal\vec{\alpha}_1^\intercal = \vec{\alpha}_1\mathbf{S}\vec{\alpha}_1^\intercal \,. \tag{4.3}$$

To compute $\vec{\alpha}_1$ we have to look for the vector that maximize such a variance.

Obviously, one can attend a value for the variance as high as wished, raising the modulo $\|\vec{\alpha}_1\| = \sqrt{\vec{\alpha}_1^T\vec{\alpha}_1}$. In order to let the maximization problem have a solution, a restriction is imposed: $\vec{\alpha}_1^T\vec{\alpha}_1 = 1$. This constrained optimization problem is handled making use of Lagrange multipliers:

$$\Lambda(\vec{\alpha}_1, \lambda) = \vec{\alpha}_1\mathbf{S}\vec{\alpha}_1^\intercal - \lambda(\vec{\alpha}_1\vec{\alpha}_1^\intercal - 1) \,, \tag{4.4}$$

and computing de partial derivative of Λ with respect to $\vec{\alpha}_1$:

$$\frac{\partial\Lambda}{\partial\vec{\alpha}_1} = 2\mathbf{S}\vec{\alpha}_1^\intercal - 2\lambda\vec{\alpha}_1^\intercal \,. \tag{4.5}$$

Thus, stationary points of Λ verify

$$\mathbf{S}\vec{\alpha}_1^\intercal = \lambda\vec{\alpha}_1^\intercal \,, \tag{4.6}$$

which implies that $\vec{\alpha}_1^\intercal$ must be an eigenvector of $\mathbf{S}$, with λ its correspondent eigenvalue. Multiplying both sides of Eq. (4.6) by $\vec{\alpha}_1$ on the left, we remark that

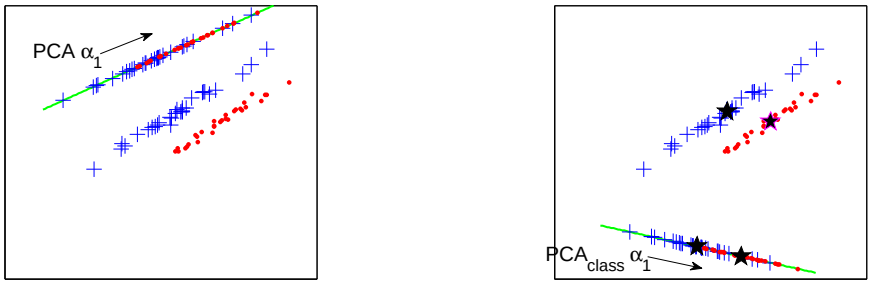
$$\vec{\alpha}_1\mathbf{S}\vec{\alpha}_1^\intercal = \lambda\vec{\alpha}_1\vec{\alpha}_1^\intercal = \lambda, \tag{4.7}$$

which means that the variance of the obtained variable $\vec{y}_1$ equals λ. For this reason $\vec{\alpha}_1$ must be the leading eigenvector of $\mathbf{S}$, the one correspond to the maximal eigenvalue.

Compute the Second and Following Principal Components The PCs following the first one are defined in an incremental fashion by choosing new directions orthogonal to those already considered and such that the sum of the projected variances over each direction is maximal. Explicitly, if we look for two PCs, *i.e.* $C = 2$, we look for a 2-dimensional variable $\vec{Y} = \begin{bmatrix}\vec{\alpha}_1 \\ \vec{\alpha}_2\end{bmatrix}\vec{X}$ such that the trace of its covariance matrix, *i.e.* the sum of variances $\mathrm{Var}(Y_1) + \mathrm{Var}(Y_2)$, is maximal. It can be shown that the same result would be obtained maximizing the so-called *generalized variance* of $\vec{Y}$, which is defined as the determinant of its covariance matrix, instead of its trace.

Consider, as in previous case, the Lagrangian of the problem

$$\Lambda = \vec{\alpha}_1\mathbf{S}\vec{\alpha}_1^\intercal + \vec{\alpha}_2\mathbf{S}\vec{\alpha}_2^\intercal - \lambda_1(\vec{\alpha}_1\vec{\alpha}_1^\intercal - 1) - \lambda_2(\vec{\alpha}_2\vec{\alpha}_2^\intercal - 1) \,, \tag{4.8}$$



(a) (b)

FIGURE 4.2: PCA: some 2-dimensional labelled data (blue crosses and red circles) projected into their 1-dimensional principal subspaces (represented by the green line). (a) classical unsupervised PCA, (b) class-oriented PCA. In (b) black stars represents the 2 classes centroids (sample means).

partial derivatives with respect to $\vec{\alpha}_1$ and $\vec{\alpha}_2$ attend zero under the following conditions:

$$\mathbf{S}\vec{\alpha}_1^\intercal = \lambda_1\vec{\alpha}_1^\intercal \tag{4.9}$$
$$\mathbf{S}\vec{\alpha}_2^\intercal = \lambda_2\vec{\alpha}_2^\intercal\,. \tag{4.10}$$

It means that $\vec{\alpha}_1^\intercal$ and $\vec{\alpha}_2^\intercal$ must be eigenvectors of $\mathbf{S}$ with correspondent eigenvalues given by λ_1 and λ_2. Moreover, as before, λ_1 and λ_2 equal the variances of variable components Y_1 and Y_2, and since the goal is maximizing the sum of such variables we have to choose $\vec{\alpha}_1$ and $\vec{\alpha}_2$ as the two leading vectors of $\mathbf{S}$.
Let us remark that the covariance between Y_1 and Y_2 is given by $\vec{\alpha}_1^T\mathbf{S}\vec{\alpha}_2$ which equals zero, since $\vec{\alpha}_1^T\vec{\alpha}_2 = 0$, by orthogonality. In particular the principal variables are uncorrelated, which is a remarkable property of the PCA.

In the general case of a C-dimensional projection space, it can be shown by induction that the PCs would corresponds to the C leading eigenvectors of the covariance matrix $\mathbf{S}$.

4.2.2 Original vs Class-Oriented PCA

The classical version of the PCA method is unsupervised. On the other hand a profiling attacker is not only provided with a set of traces $(\vec{x})_{i=1..N}$, but he also has the knowledge of the target values handled during each acquisition. We denote $(\vec{x}_i, z_i)_{i=1..N_p}$ the labelled set of traces . In Fig. 4.2 the same toys data of Fig. 4.1 have been grouped into 2 classes. Even if before projection the two groups are clearly separable, projecting data over the first principal component given by the classical PCA algorithm 4.2(a), such a separability is lost. In the supervised context, and for the sake of distinguishing the target value assumed by the target Z in new executions, the idea of the *Class-Oriented* PCA is to consider *equivalent* all traces belonging to the same class. Modelling traces of a same class as traces showing the same characteristic form plus a random noise, the denoised characteristic form can be estimated by

the sample means of the traces in the class. Let

$$\overline{\vec{x}}^z = \frac{1}{N_z} \sum_{i:\ z_i = z} \vec{x}_i$$

be the empirical mean of traces belonging to the same class z, being N_z be the number of traces belonging to class z. The class-oriented version of the PCA consists in applying the PCA dimensionality reduction to the set $(\overline{\vec{x}}^z)_{z \in \mathcal{Z}}$, instead of applying it directly to the traces $\vec{x}_i$. This implies that the empirical covariance matrix will be computed using only the $|\mathcal{Z}|$ average traces. Equivalently, in case of *balanced* acquisitions (N_z constant for each class z), it amounts to replace the covariance matrix $\mathbf{S}$ of data in (4.1) by the so-called *between-class* or *inter-class scatter matrix*, given by:

$$\mathbf{S_B} = \sum_{z \in \mathcal{Z}} N_z (\overline{\vec{x}}^z - \overline{\vec{x}})(\overline{\vec{x}}^z - \overline{\vec{x}})^\intercal \ . \tag{4.11}$$

Remark that $\mathbf{S_B}$ coincides, up to a multiplicative factor, to the covariance matrix obtained using the class-averaged traces. In this way we focus the attention on information that discriminate the characteristic forms of classes, *i.e.* target values. Figure 4.2(b) shows how the 2-class toy data are projected over the first class-oriented principal component: in the figure, black stars represent the class sample means. Projected data are kept slightly better separated than in Fig. 4.2(a).

4.2.3 Computational Consideration

Performing PCA (and LDA, as explained later) always asks to compute the eigenvector of some symmetric matrix $\mathbf{S}$, obtained by the multiplication of a constant, a matrix and the same matrix transposed, *e.g.* $\mathbf{S} = \frac{1}{N}\mathbf{M}\mathbf{M}^\intercal$. Let $\mathbf{M}$ have dimension $D \times N$, and suppose $N \ll D$. This case is almost always attended when performing class-oriented PCA, because in such a case N corresponds to the number of classes $|\mathcal{Z}|$. Anyway, for high-dimensional data, *i.e.* D high, it can be attended even when performing classical PCA. Thus, in such a common case, the $D \times D$ matrix $\mathbf{S}$ is far from being a full-rank matrix, since $\mathrm{rank}(\mathbf{S}) \leq N \ll D$. That is why we expect to find a most N eigenvectors; often columns of $\mathbf{M}$ are linearly dependent, because for example they are forced to have zero mean, so actually the rank of $\mathbf{S}$ is strictly less than N and we expect to obtain at most $N - 1$ eigenvectors.

A practical problem in case of high-dimensional data, is represented by the computation and the storage of the $D \times D$ matrix $\mathbf{S}$. This problem can be bypassed exploiting the following lemma coming from linear algebra, as proposed by Archambeau *et al.* [Arc+06]:

Lemma 1. For any $D \times N$ matrix $\mathbf{M}$, mapping $\vec{x} \mapsto \mathbf{M}\vec{x}$ is a one-to-one mapping that maps eigenvectors of $\mathbf{M}^\intercal\mathbf{M}$ ($N \times N$) onto those of $\mathbf{M}\mathbf{M}^\intercal$ ($D \times D$).

This lemma allows to compute and store the smaller $N \times N$ matrix $\tilde{\mathbf{S}} = \frac{1}{N}\mathbf{M}^\intercal\mathbf{M}$, compute its $(N \times 1)$-sized eigenvectors $\vec{\beta}_i$ and the relative eigenvalues λ_i, and then convert them into eigenvectors of $\mathbf{S}$, given by $\vec{\alpha}_i = \mathbf{M}\vec{\beta}_i$. Observing that by definition $\tilde{\mathbf{S}}\vec{\beta}_i = \frac{1}{N}\mathbf{M}^\intercal\mathbf{M}\vec{\beta}_i = \lambda_i\vec{\beta}_i$ the lemma is easy to verify:

$$\mathbf{S}\vec{\alpha}_i = \frac{1}{N}\mathbf{M}\mathbf{M}^\intercal\mathbf{M}\vec{\beta}_i = \lambda_i\mathbf{M}\vec{\beta}_i = \lambda_i\vec{\alpha}_i \ . \tag{4.12}$$

It is not guaranteed that eigenvectors $\vec{\alpha}_i$ obtained in this way have norm equal to 1. This is why a normalization step should follow it we aim to obtain an orthonormal set of eigenvectors.

4.2.4 The Choice of the Principal Components

The introduction of the PCA method in SCA context (either in its classical or class-oriented version) has raised some non-trivial questions: *how many* principal components and *which ones* are sufficient/necessary to reduce the trace size (and thus the attack processing complexity) without losing important discriminative information?

Until 2015, an answer to the questions above was given in [CK14b], linked to the concept of *explained variance* (or *explained global variance*, EGV for short) of a PC $\vec{\alpha}_i$:

$$\mathrm{EGV}(\vec{\alpha}_i) = \frac{\lambda_i}{\sum_{k=1}^{r} \lambda_k} , \tag{4.13}$$

where r is the rank of the covariance matrix $\mathbf{S}$, and λ_j is the eigenvalue associated to the j-th PC $\vec{\alpha}_j$. $\mathrm{EGV}(\vec{\alpha}_i)$ is the variance of the data projected over the i-th PC (which equals λ_i) divided by the total variance of the original data (given by the trace of the covariance matrix $\mathbf{S}$, *i.e.* by the sum of all its non-zero eigenvalues). By definition of EGV, the sum of all the EGV values is equal to 1; for this reason this quantity is often multiplied by 100 and expressed as percentage. Exploiting the EGV to choose among the PCs consists in fixing a wished *cumulative explained variance* β and in keeping C different PCs, where C is the minimum integer such that

$$\mathrm{EGV}(\vec{\alpha}_1) + \mathrm{EGV}(\vec{\alpha}_2) + \cdots + \mathrm{EGV}(\vec{\alpha}_C) \geq \beta \, . \tag{4.14}$$

However, if the attacker has a constraint for the reduced dimension C, the EGV notion simply suggests to keep the first C components, taking for granted that the optimal way to chose PCs is in their natural order. This assumption is not always confirmed in SCA context: in some works, researchers have already remarked that the first components sometimes contain more noise than information [BHW12; Spe+15] and it is worth discarding them. For the sake of providing a first example of this behaviour on publicly accessible traces, we applied a class-oriented PCA on 3000 traces from the DPA contest v4 [Par]; we focused over a small 1000-dimensional window in which, in complete knowledge about masks and other countermeasures, information about the first Sbox processing leaks (during the first round). In Fig. 4.3 the first and the sixth PCs are plotted. It may be noticed that the first component indicates that one can attend a high variance by exploiting the regularity of the traces, given by the clock signal, while the sixth one has high coefficients localised in a small time interval, very likely to signalize the instants in which the target sensitive variable leaks.

A single method adapted to SCA context has been proposed until 2015 to automatically choose PCs [Mav+12] while dealing with the issue raised in Fig. 4.3. It was based on the following assumption:

Assumption 1. The leaking side-channel information is localised in few points of the acquired trace.

Such an assumption 1 is reasonable in SCA contexts where the goal of the security developers is to minimize the number of leaking points. Under this assumption, the authors of [Mav+12] use for side-channel attack purposes the *Inverse Participation Ratio* (IPR), a measure widely exploited in Quantum Mechanics domain (see

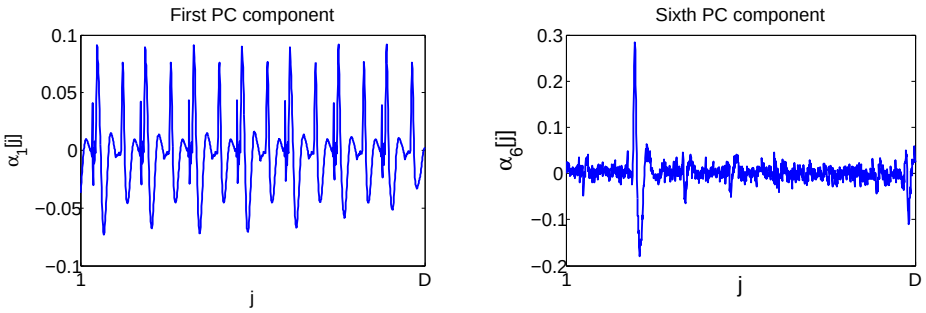


FIGURE 4.3: First and sixth PCs in DPA contest v4 trace set (between time samples 198001 and 199000)

for example [**guhr1998random**]). They propose to use such a score to evaluate the eigenvectors *localization*. It is defined as follows:

$$\mathrm{IPR}(\vec{\alpha}_i) = \sum_{j=1}^{D} \vec{\alpha}_i[j]^4 \,. \tag{4.15}$$

The authors of [Mav+12] suggest to collect the PCs in decreasing order with respect to the IPR score.

The selection methods provided by the evaluation of the EGV and of the IPR are somehow complementary: the former is based only on the eigenvalues associated to the PCs and does not consider the form of the PCs themselves; the latter completely discards the information given by the eigenvalues of the PCs, considering only the distribution of their coefficients. In the next section we describe a new method, part of the contributions published in [**cagli2015enhancing**], that builds a bridge between the EGV and the IPR approaches. As we will argue, our method, based on the so-called *explained local variance*, does not only lead to the construction of a new selection criterion, but also permits to modify the PCs, choosing individually the coefficients to keep and those to discard.

4.2.4.1 Explained Local Variance Selection Method

The method we develop in this section is based on a compromise between the variance provided by each PC (more precisely its EGV) and the number of time samples necessary to achieve a consistent part of such a variance. To this purpose we introduce the concept of *Explained Local Variance* (ELV). Let us start by giving some intuition behind our new concept. Thinking to the observations $\vec{x}^\intercal$, or to the class-averages $\overline{\vec{x}}^\intercal$ in class-oriented PCA case, as realizations of a random variable $\vec{X}^\intercal$, we

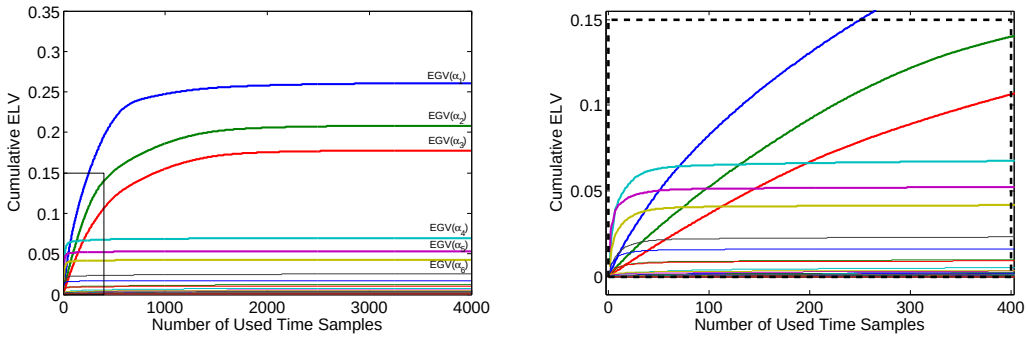


FIGURE 4.4: Cumulative ELV trend of principal components. On the right a zoom of the plot on the left. Data acquisition described in Sec. 4.4.

have that λ_i is an estimator for the variance of the random variable $\vec{X}^\intercal \cdot \vec{\alpha}_i$. Developing, we obtain

$$\lambda_i = \hat{\mathrm{Var}}(\sum_{j=1}^{D} \vec{X}^\intercal[j]\vec{\alpha}_i[j]) = \sum_{j=1}^{D}\sum_{k=1}^{D} \hat{\mathrm{Cov}}(\vec{X}^\intercal[j]\vec{\alpha}_i[j], \vec{X}^\intercal[k]\vec{\alpha}_i[k]) = \tag{4.16}$$

$$= \sum_{j=1}^{D} \vec{\alpha}_i[j] \sum_{k=1}^{D} \vec{\alpha}_i[k] \hat{\mathrm{Cov}}(\vec{X}^\intercal[j], \vec{X}^\intercal[k]) = \sum_{j=1}^{D} \vec{\alpha}_i[j](\mathbf{S}_j^\intercal \cdot \vec{\alpha}_i) = \tag{4.17}$$

$$= \sum_{j=1}^{D} \vec{\alpha}_i[j]\lambda_i\vec{\alpha}_i[j] = \sum_{j=1}^{D} \lambda_i \vec{\alpha}_i[j]^2 \tag{4.18}$$

where $\mathbf{S}_j^\intercal$ denotes the j-th row of $\mathbf{S}$ and (4.18) is justified by the fact that $\vec{\alpha}_i$ is an eigenvector of $\mathbf{S}$, with λ_i its corresponding eigenvalue. The result of this computation is quite obvious, since $\| \vec{\alpha}_i \| = 1$, but it evidences the contribution of each time sample in the information held by the PC. This makes us introduce the following definition:

Definition 1. The *Explained Local Variance* of a PC $\vec{\alpha}_i$ in a sample j, is defined by

$$\mathrm{ELV}(\vec{\alpha}_i, j) = \frac{\lambda_i \vec{\alpha}_i[j]^2}{\sum_{k=1}^{r} \lambda_k} = \mathrm{EGV}(\vec{\alpha}_i)\vec{\alpha}_i[j]^2 \,. \tag{4.19}$$

Let $\mathcal{J} = \{j_1^i, j_2^i, \ldots, j_D^i\} \subset \{1, 2, \ldots, D\}$ be a set of indexes sorted such that $\mathrm{ELV}(\vec{\alpha}_i, j_1^i) \geq \mathrm{ELV}(\vec{\alpha}_i, j_2^i) \geq \cdots \geq \mathrm{ELV}(\vec{\alpha}_i, j_D^i)$. It may be observed that the sum over all the $\mathrm{ELV}(\vec{\alpha}_i, j)$, for $j \in [1, \ldots, D]$, equals $\mathrm{EGV}(\vec{\alpha}_i)$. If we operate such a sum in a cumulative way following the order provided by the sorted set $\mathcal{J}$, we obtain a complete description of the trend followed by the component $\vec{\alpha}_i$ to achieve its EGV. As we can see in Fig. 4.4, where such cumulative ELVs are represented, the first 3 components are much slower in achieving their final EGV, while the 4^{th}, the 5^{th} and the 6^{th} achieve a large part of their final EGVs very quickly (*i.e.* by adding the ELV contributions of much less time samples). For instance, for $i = 4$, the sum of the $\mathrm{ELV}(\vec{\alpha}_4, j_k^4)$, with $k \in [1, \ldots, 30]$, almost equals $\mathrm{EGV}(\vec{\alpha}_4)$, whereas the same sum for $i = 1$ only achieves about the 15% of $\mathrm{EGV}(\vec{\alpha}_1)$. Actually, the EGV of the 4^{th}, the 5^{th} and the 6^{th} component only essentially depends on a very few time samples. This observation, combined with Assumption 1, suggests that they are more suitable for SCA than the three first ones. To validate this statement, it suffices to look at the

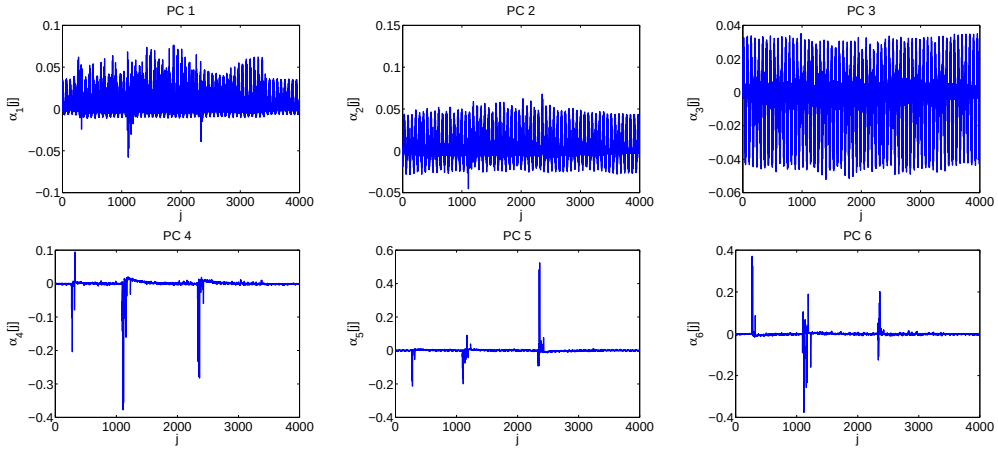


FIGURE 4.5: The first six PCs. Acquisition campaign on an 8-bit AVR Atmega328P (see Sec. 4.4).

form of such components (Fig. 4.5): the leading ones are strongly influenced by the clock, while the latest ones are well localised over the leaking points.

Operating a selection of components *via* ELV, in analogy with the EGV, requires to fix the reduced space dimension C, or a threshold β for the cumulative ELV. In the first case, the maximal ELVs of each PC are compared, and the C components achieving the highest values of such ELVs are chosen. In the second case, all pairs (PC, time sample) are sorted in decreasing order with respect to their ELV, and summed until the threshold β is achieved. Then only PCs contributing in this sum are selected.

We remark that the ELV is a score associated not only to the whole components, but to each of their coefficients. This interesting property can be exploited to further remove, within a selected PC, the non-significant points, *i.e.* those with a low ELV. In practice this is done by setting these points to zero. That is a natural way to exploit the ELV score in order to operate a kind of *denoising* for the reduced data, making them only depend on the significant time samples. In Sec. 4.4 (scenario 4) we test the performances of an attack varying the number of time samples involved in the computation of the reduced data, and showing that such a denoising processing might impact significantly.

4.3 Linear Discriminant Analysis

4.3.1 Fisher's Linear Discriminant and Terminology Remark

Fisher's Linear Discriminant [Fuk90] is another statistical tool for dimensionality reduction, which is commonly used as a preliminary step before classification. Indeed it seeks for linear combinations of data that characterize or separate two or more classes, not only spreading class centroids as much as possible, like the class-oriented PCA does, but also minimizing the so-called *intra-class variance*, i.e. the variance shown by data belonging to the same class. The terms Fisher's Linear Discriminant and Linear Discriminant Analysis (LDA) are often used interchangeably, and in particular in SCA literature the Fisher's Linear Discriminant is almost always referred to as LDA, *e.g.* [Bru+ a; SA08]. As we anticipated in Chapter 3 - Example 3.1.3, this widely-accepted abuse is due to the fact that under the assumptions leading to

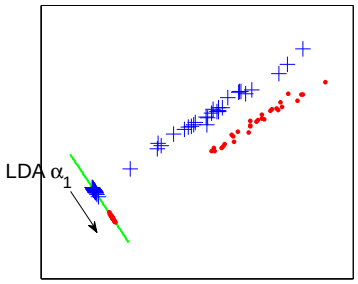


FIGURE 4.6: LDA: some 2-dimensional labelled data (blue crosses and red circles) projected onto their 1-dimensional discriminant component (represented by the green line).

the LDA classification tools (*i.e.* Gaussian class-conditional densities, sharing a common covariance matrix), the solution provided by the Fisher's Linear Discriminant (that does not require such assumptions) is the same as the solution provided by the LDA. From now on we will use the more common terminology LDA to refer to Fisher's Linear Discriminant.

4.3.2 Description

Applying LDA consists in maximizing the so-called *Rayleigh quotient*:

$$\vec{\alpha}_1 = \text{argmax}_{\vec{\alpha}} \frac{\vec{\alpha}^\intercal \mathbf{S_B} \vec{\alpha}}{\vec{\alpha}^\intercal \mathbf{S_W} \vec{\alpha}} , \tag{4.20}$$

where $\mathbf{S_B}$ is the *between-class scatter matrix* already defined in (4.11) and $\mathbf{S_W}$ is called *within-class* (or intra-class) *scatter matrix*:

$$\mathbf{S_W} = \sum_{z \in \mathcal{Z}} \sum_{i=1}^{N_p} (\vec{x}_i - \overline{\vec{x}}^z)(\vec{x}_i - \overline{\vec{x}}^z)^\intercal . \tag{4.21}$$

Remark 4.1. Let $\mathbf{S}$ be the the global covariance matrix of data, also called *total scatter matrix*, defined in (4.1); we have the following relationship between $\mathbf{S_W}$, $\mathbf{S_B}$ and $\mathbf{S}$:

$$\mathbf{S} = \frac{1}{N_p} (\mathbf{S_W} + \mathbf{S_B}) . \tag{4.22}$$

It can be shown that the vector $\vec{\alpha}_1$ which maximizes (4.20) must satisfy $\mathbf{S_B}\vec{\alpha}_1 = \lambda \mathbf{S_W}\vec{\alpha}_1$, for a constant λ, *i.e.* has to be an eigenvector of $\mathbf{S_W}^{-1}\mathbf{S_B}$. Moreover, for any eigenvector $\vec{\alpha}$ of $\mathbf{S_W}^{-1}\mathbf{S_B}$, with associated eigenvalue λ, the Rayleigh quotient equals such a λ:

$$\frac{\vec{\alpha}^\intercal \mathbf{S_B} \vec{\alpha}}{\vec{\alpha}^\intercal \mathbf{S_W} \vec{\alpha}} = \lambda . \tag{4.23}$$

Then, among all eigenvectors of $\mathbf{S_W}^{-1}\mathbf{S_B}$, $\vec{\alpha}_1$ must be the leading one.

The computation of the eigenvectors of $\mathbf{S_W}^{-1}\mathbf{S_B}$ is known under the name of *generalized eigenvector problem*. The difficulty here comes from the fact that $\mathbf{S_W}^{-1}\mathbf{S_B}$ is not

guaranteed to be symmetric. Due to this non-symmetry, $\vec{\alpha}_1$ and the others eigenvectors do not form an orthonormal basis for $\mathbb{R}^D$, but they are anyway useful for classifications scopes. Let us refer to them as *Linear Discriminant Components* (LDCs for short); as for PCs we consider them sorted in decreasing order with respect to their associated eigenvalue, which gives a score for their informativeness, see (4.23). Analogously to the PCA, the LDA provides a natural dimensionality reduction: one can project the data over the C first LDCs. In Fig. 4.6 the 2-class toy data used as example above, projected over their leading discriminant component, are depicted. The two classes are kept well separated in the 1-dimensional subspace. As for PCA, this choice might not be optimal when applying this reduction to side-channel traces. For the sake of comparison, we test in Sec. 4.4 all the selection methods proposed for the PCA (EGV, IPR and ELV) in association to the LDA as well.

In the following subsection we will present a well-known problem that affects the LDA in many practical contexts, and describe four methods that circumvent such a problem, with the intention to test them over side-channel data.

4.3.3 The Small Sample Size Problem

In the special case in which the matrix $\mathbf{S_B}$ is invertible, the generalized eigenvalue problem is convertible in a regular one, as in [SA08]. On the contrary, when $\mathbf{S_B}$ is singular, the simultaneous diagonalization is suggested to solve such a problem [Fuk90]. In this case one can take advantage by the singularity of $\mathbf{S_B}$ to apply the computational trick described in Sec. 4.2.3, since at most $r = \mathrm{rank}(\mathbf{S_B})$ eigenvectors can be found.

If the singularity of $\mathbf{S_B}$ does not affect the LDA dimensionality reduction, we cannot say the same about the singularity of $\mathbf{S_W}$: SCA and Pattern Recognition literatures point out the same drawback of the LDA, known as the *Small Sample Size problem* (SSS for short). It occurs when the total number of acquisitions N_p is less than or equal to the size D of them. The direct consequence of this problem is the singularity of $\mathbf{S_W}$ and the non-applicability of the LDA.

If the LDA has been introduced relatively lately in the SCA literature, the Pattern Recognition community looks for a solution to the SSS problem at least since the early nineties. We browsed some of the proposed solutions and chose some of them to introduce, in order to test them over side channel traces.

4.3.3.1 Fisherface Method

The most popular among the solutions to SSS is the so-called *Fisherface* method[1] [BHK97]. It simply relies on the combination between PCA and LDA: a standard PCA dimensionality reduction is performed to data, making them pass from dimension D to dimension $N_p - |\mathcal{Z}|$, which is the general maximal rank for $\mathbf{S_W}$. In this reduced space, $\mathbf{S_W}$ is very likely to be invertible and the LDA therefore applies.

[1] The name is due to the fact that it was proposed and tested for face recognition scopes.

4.3.3.2 $\mathbf{S_W}$ Null Space Method

It has been introduced by Chen et al. in [Che+00] and exploits an important result of Liu et al. [LCY92] who showed that Fisher's criterion (4.20) is equivalent to:

$$\vec{\alpha}_1 = \text{argmax}_{\vec{\alpha}} \frac{\vec{\alpha}^\intercal \mathbf{S_B} \vec{\alpha}}{\vec{\alpha}^\intercal \mathbf{S_W} \vec{\alpha} + \vec{\alpha}^\intercal \mathbf{S_B} \vec{\alpha}} \,. \tag{4.24}$$

The authors of [Che+00] point out that such a formula is upper-bounded by 1, and that it achieves its maximal value, *i.e.* 1, if and only if $\vec{\alpha}$ is in the null space of $\mathbf{S_W}$. Thus they propose to first project data onto the null space of $\mathbf{S_W}$ and then to perform a PCA, *i.e.* to select as LDCs the first $|\mathcal{Z}|-1$ eigenvectors of the between-class scatter matrix of data into this new space. More precisely, let $Q = [\vec{v}_1, \ldots, \vec{v}_{D-\text{rank}(\mathbf{S_W})}]$ be the matrix of vectors that span the null space of $\mathbf{S_W}$. [Che+00] proposes to transform the data $\vec{x}$ into $\vec{x}' = QQ^\intercal \vec{x}$. Such a transformation maintains the original dimension D of the data, but let the new within-class matrix $\mathbf{S'_W} = QQ^\intercal \mathbf{S_W} QQ^\intercal$ be the null $D \times D$ matrix. Afterwards, the method looks for the eigenvectors of the new between-class matrix $\mathbf{S'_B} = QQ^\intercal \mathbf{S_B} QQ^\intercal$. Let U be the matrix containing its first $|\mathcal{Z}|-1$ eigenvectors: the LDCs obtained via the $\mathbf{S_W}$ null space method are the columns of $QQ^\intercal U$.

4.3.3.3 Direct LDA

As the previous, this method, introduced in [YY01], privileges the low-ranked eigenvectors of $\mathbf{S_W}$, but proposes to firstly project the data onto the rank space of $\mathbf{S_B}$, arguing the fact that vectors of the null space of $\mathbf{S_B}$ do not provide any between-class separation of data. Let $D_B = V^\intercal \mathbf{S_B} V$ be the diagonalization of $\mathbf{S_B}$, and let $V^\star$ be the matrix of the eigenvectors of $\mathbf{S_B}$ that are not in its null space, *i.e.* whose eigenvalues are different from zero. Let also $D_B^\star$ denotes the matrix $V^{\star\intercal} \mathbf{S_B} V^\star$; transforming the data $\vec{x}$ into ${D_B^\star}^{1/2} V^{\star\intercal} \vec{x}$ makes the between-class variance to be equal to the $(|\mathcal{Z}|-1) \times (|\mathcal{Z}|-1)$ identity matrix. After this transformation the within-class variance assumes the form $\mathbf{S'_W} = {D_B^\star}^{1/2} V^{\star\intercal} \mathbf{S'_W} V^\star {D_B^\star}^{1/2}$. After storing the C lowest-rank eigenvectors in a matrix $U^\star$, the LDCs obtained via the Direct LDA method are the columns of $V^\star {D_B^\star}^{1/2} U^{\star\intercal}$.

4.3.3.4 $\mathbf{S_T}$ Spanned Space Method

The last variant of LDA that we consider has been proposed in [Hua+02] and is actually a variant of the Direct LDA: instead of removing the null space of $\mathbf{S_B}$ as first step, this method removes the null space of $\mathbf{S_T} = \mathbf{S_B} + \mathbf{S_W}$. Then, denoting $\mathbf{S'_W}$ the within-class matrix in the reduced space, the reduced data are projected onto its null space, i.e. are multiplied by the matrix storing by columns the eigenvectors of $\mathbf{S'_W}$ associated to the null eigenvector, thus reduced again. A final optional step consists in verifying whether the between-class matrix presents a non-trivial null-space after the last projection and, in this case, in effectuating a further projection removing it.

Remark 4.2. Let us remark that, from a computational complexity point of view (see [Hua+02] for a deeper discussion), the least time-consuming procedure among the four proposed is the Direct LDA, followed by the Fisherface and the $\mathbf{S_T}$ Spanned Space Method, that have a similar complexity. The $\mathbf{S_W}$ Null Space Method is in general much more expensive, because the task of removing the $\mathbf{S_W}$ null space requires the actual computation of the $(D \times D)$-dimensional matrix $\mathbf{S_W}$, *i.e.* the computational trick described in Sec. 4.2.3 is not applicable. On the contrary, the other three methods take advantage of such a procedure, reducing drastically their complexity.

4.4 Experimental Results

In this section we compare the different extractors (*i.e.* functions applying a data dimensionality reduction, see Sec. 2.2.4) provided by the PCA and the LDA in association with the different techniques of components selection. Defining an universal criterion to compare the different extractors would not make sense since the latter one should encompass a lot of parameters, sometimes opposite, that vary according to the context (amount of noise, specificity of the information leakage, nature of the side channel, etc.). For this reason we choose to split our comparisons into four scenarios. Each scenario has a single varying parameter that, depending on the attacker context, may wish to be minimized. Hereafter the definition of the four scenarios:

[Scenario 1] varying parameter: number of attack traces N_a,

[Scenario 2] varying parameter: number of profiling traces N_p,

[Scenario 3] varying parameter: number of projecting components selected C,

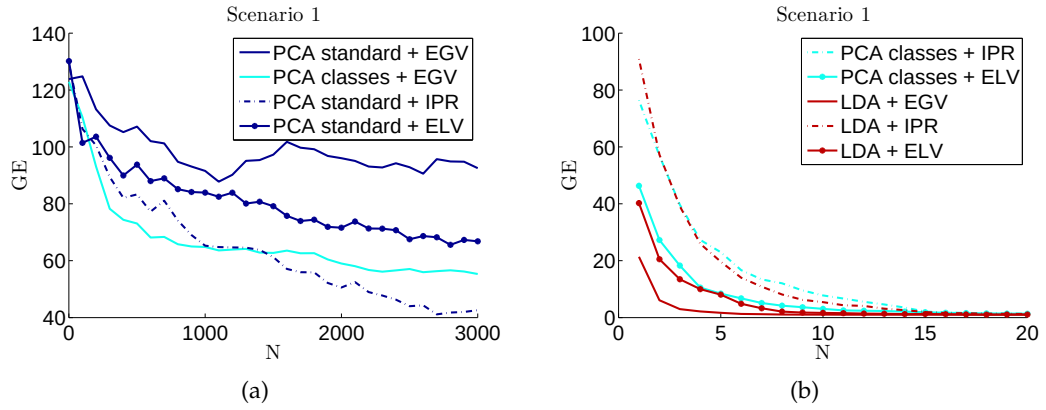
[Scenario 4] varying parameter: number of original time samples implied into the trace preprocessing computation $\sharp\text{PoI}$.

For scenarios in which N_p is fixed, the value of N_p is chosen high enough to avoid the SSS problem, and the extensions of LDA presented in Sec. 4.3.3 are not evaluated. This choice of N_p will imply that the LDA is always performed in a favourable situation, which makes expect the LDA to be particularly efficient for these experiments. Consequentely, for the scenarios in which N_p is high, our goal is to study whether the PCA can be made almost as efficient as the LDA thanks to the component selection methods discussed in Sec. 4.2.4.

4.4.0.1 The testing adversary.

Our testing adversary attacks an 8-bit AVR microprocessor Atmega328P and acquires power-consumption traces via the ChipWhisperer platform [OC14].[2] The target device stores a secret 128-bit key and performs the first steps of an AES: the loading of 16 bytes of the plaintext, the AddRoundKey step and the AES Sbox. It has been programmed twice: two different keys are stored in the device memory during the acquisition of the profiling and of the attack traces, to simulate the situation of two identical devices storing a different secret. The size D of the traces equals 3996. The sensitive target variable is the output of the first Sbox processing, but, since the key is fixed also during the profiling phase, and both Xor and Sbox operations are bijective, we expect to detect three interesting regions (as those high-lighted by PCs 4, 5 and 6, in Fig. 4.5): the reading of the first byte of the plaintext, the first AddRoundKey and the first Sbox. We consider an *identity classification* leaking function (i.e. we make minimal assumption on the leakage function, see Sec. 2.2.1.2), which implies that the 256 possible values of the Sbox output yields to 256 classes. For each class we assume that the adversary acquires the same number N of traces, *i.e.* $N_p = N \times 256$. After the application of the extractor ϵ, the trace size is reduced to C. Then the attacker performs a Template Attack (see Sec. 2.2.3.1), using C-variate Gaussian templates.

[2] This choice has been done to allow for reproducibility of the experiments.



(a) (b)

FIGURE 4.7: Guessing Entropy as function of the number of attack traces for different extraction methods. All Guessing Entropies are estimated as the average rank of the right key over 100 independent experiments.

4.4.0.2 Scenario 1.

To analyse the dependence between the extraction methods presented in Sections 4.2 and 4.3 and the number of attack traces N_a needed to achieve a given GE (Guessing Entropy, see Sec. 2.2.2), we fixed the other parameters as follows: $N = 50$ ($N_p = 50 \times 256$), $C = 3$ and $\sharp\text{PoI} = 3996$ (all points are allowed to participate in the building of the PCs and of the LDCs). The experimental results, depicted in Fig. 4.7(a)-(b), show that the PCA standard method has very bad performances in SCA, while the LDA outperforms the others. Concerning the class-oriented PCA, we observe that its performance is close to that of LDA when combined with the selection methods ELV (which performs best) or IPR, while it is similar to the one of classic PCA when associated with the EGV selection.

4.4.0.3 Scenario 2.

Now we test the behaviour of the extraction methods as function of the number N_z of available profiling traces per class. The number of components C is still fixed to 3, $\sharp\text{PoI} = 3996$ again and the number of attack traces is $N_a = 100$. This scenario has to be divided into two parts: if $N_z \leq 15$, then $N_p < D$ and the SSS problem occurs. Thus, in this case we test the four extensions of LDA presented in Sec. 4.3.3, associated to either the standard selection, to which we abusively refer as EGV,[3] or to the IPR selection. We compare them to the class-oriented PCA associated to EGV, IPR or ELV. The ELV selection is not performed for the techniques extending LDA, since for some of them the projecting LDCs are not associated to some eigenvalues in a meaningful way. On the contrary, if $N \geq 16$ there is no need to approximate the LDA technique, so the classical one is performed. Results for this scenario are shown in Fig. 4.8. It may be noticed that the combinations class-oriented PCA + ELV/IPR select exactly the same components, for our data, see Fig. 4.8(e) and do not suffer from the lack of profiling traces. They are slightly outperformed by the $\mathbf{S_W}$ Null Space method associated with the EGV, see Fig.4.8(d). The Direct LDA (Fig. 4.8(b)) method also provides a good alternative, while the other tested methods do not show a stable behaviour. The results in absence of the SSS problem (Fig.4.8(f))

[3] It consists in keeping the C first LDCs (the C last for the Direct LDA)

confirm that the standard PCA is not adapted to SCA, even when provided with more profiling traces. It also shows that among class-oriented PCA and LDA, the class-oriented PCA converges faster.

4.4.0.4 Scenario 3.

Let C be now variable and let the other parameters be fixed as follows: $N_a = 100, N_z = 200, \sharp\text{PoI} = 3996$. Looking at Fig. 4.9, we might observe that the standard PCA might actually well perform in SCA context if provided with a larger number of kept components; on the contrary, a little number of components suffices to the LDA. Finally, keeping more of the necessary components does not worsen the efficiency of the attack, which allows the attacker to choose C as the maximum value supported by his computational means.

Remark 4.3. In our experiments the ELV selection method only slightly outperforms the IPR. Nevertheless, since it relies on more sound and more general observations, *i.e.* the maximization of explained variance concentrated into few points, it is likely to be more robust and less case-specific. For example, in Fig. 4.8(f) it can be remarked that while the class-oriented PCA + ELV line keeps constant on the value 1 of guessing entropy, the class-oriented PCA + IPR is sometimes higher than 1.

4.4.0.5 Scenario 4.

This is the single scenario in which we allow the ELV selection method to not only select the components to keep but also to modify them, keeping only some coefficients within each component, setting the other ones to zero. We select pairs *(component, time sample)* in decreasing order of the ELV values, allowing the presence of only $C = 3$ components and $\sharp\text{PoI}$ different times samples, *i.e.* each component must have only 3 entries different from 0. Looking at Fig. 4.10 we might observe that the LDA allows to achieve the maximal guessing entropy with only 1 PoI in each of the 3 selected components. Actually, adding PoIs worsen its performances, which is coherent with the assumption that the vulnerable information leaks in only a few points. Such points are excellently detected by the LDA. Adding contribution from other points raises the noise, which is then compensated by the contributions of further noisy points, in a very delicate balance. Such a behaviour is clearly visible in standard PCA case: the first 10 points considered raise the level of noise, that is then balanced by the last 1000 points.

4.4.1 Overview of this Study and Conclusions

This study focused on two well-known techniques to construct extractors for side-channel traces, the PCA and the LDA. The LDA method is more adequate than the PCA one, thanks to its class-distinguishing asset, but more expensive and not always available in concrete situations. We deduced from a general consideration about side-channel traces, *i.e.* the fact that for secured implementations the vulnerable leakages are concentrated into few points, a new methodology for selecting components, called ELV. We showed that the class-oriented PCA, equipped with the ELV, achieves performances close to those of the LDA, becoming a cheaper and valid alternative to the LDA. Being our core consideration very general in side-channel context, we believe that our results are not case-specific.
A second part of the proposed study analysed experimentally some alternatives to the LDA in presence of SSS problem proposed in Pattern Recognition literature.

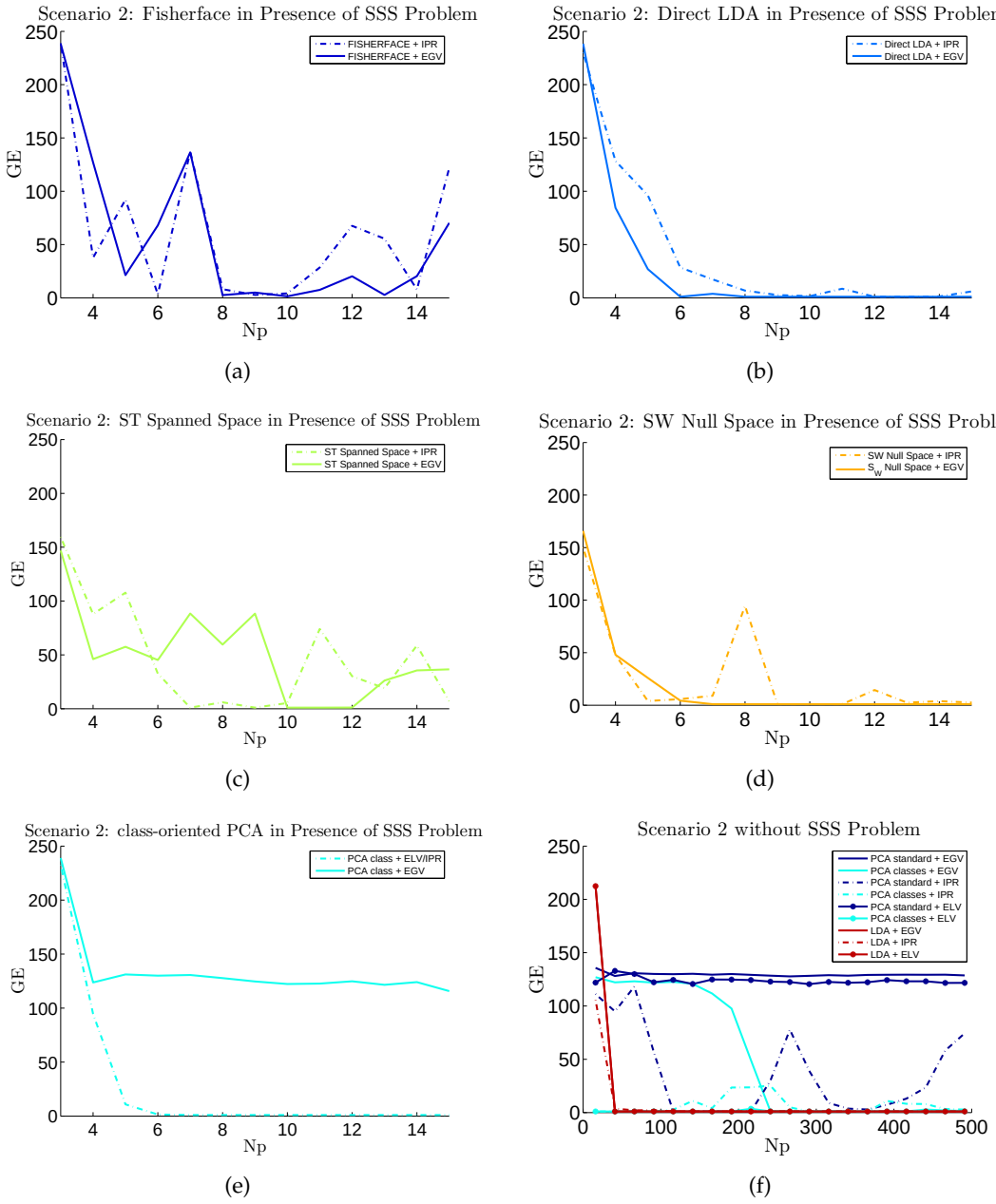


FIGURE 4.8: Guessing Entropy as function of the number of profiling traces. Figures (a)-(d): methods extending the LDA in presence of SSS problem; Figure (e): class-oriented PCA in presence of the SSS problem; Figure (f): number of profiling traces high enough to avoid the SSS problem.

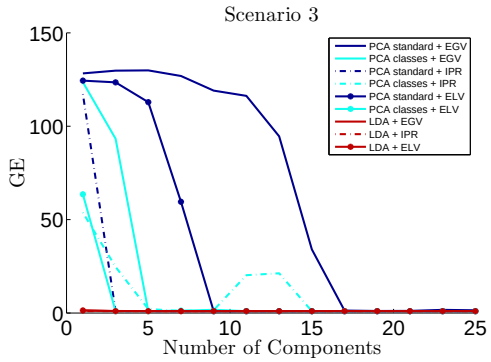


FIGURE 4.9: Guessing Entropy as function of the trace size after reduction.

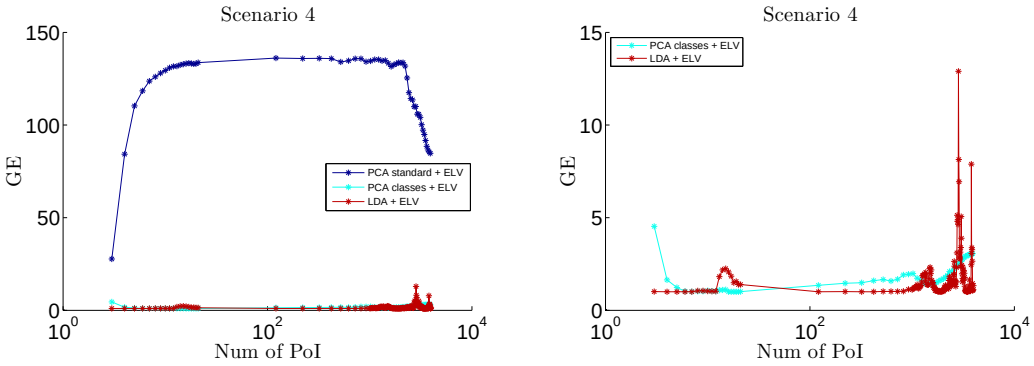


FIGURE 4.10: Guessing Entropy as function of the number of time samples contributing to the extractor computation.

Such experiments showed that the Direct LDA and the $\mathbf{S_W}$ Null Space Method are promising techniques, exhibiting performances close to the ones given by the ELV-equipped class-oriented PCA. A synthetic overview of the performed comparisons in scenarios 1,2 and 3 is depicted in Table 4.1.

4.5 Misaligning Effects

The fact that trace misalignment leads to a drastic drop of the *dimensionality reduction/ template attack* routine is well-known. When we are in presence of a misalignment, caused by the implementation of a countermeasure or by the lack of a good trigger signal for the acquisition, the application of some previous re-synchronization techniques is recommended (see for instance [CK14c], where the same PCA and LDA techniques are exploited as template attack preprocessing, after a prior resnchronization). In this section we experimentally show how the approach based on linear dimensionality reduction described in this chapter is affected by traces misalignment. To this aim, we simply take the same data and parameters exploited for Scenario 1 in Sec. 4.4, and artificially misalign them through the technique proposed in Appendix B with parameters `sigma`$= 6$, `B`$= 4$. Then we tried to attack them through the 9 methodologies tested in Scenario 1. It may be noticed in Fig. 4.11 that none of the 9 techniques is still efficient, included the optimal LDA+EGV that lead to minimal guessing entropy with the synchronized traces using less than 7 attack traces. In this case it cannot lead to successful attack in less than 3000 traces.

		Parameter to minimize			
Method	Selection	N	N' (SSS)	N' ($\neg$SSS)	C
PCA standard	EGV	-		-	-
PCA standard	ELV	-		-	-
PCA standard	IPR	-		-	+
PCA class	EGV	-	-	-	-
PCA class	ELV	+	★	★	+
PCA class	IPR	+	★	+	-
LDA	EGV	★		+	★
LDA	ELV	+		+	★
LDA	IPR	+		+	★
$\mathbf{S_W}$ Null Space	EGV		★		
$\mathbf{S_W}$ Null Space	IPR		+		
Direct LDA	EGV		★		
Direct LDA	IPR		+		
Fisherface			-		
$\mathbf{S_T}$ Spanned Space			-		

TABLE 4.1: Overview of extractors performances in tested situations. Depending on the adversary purpose, given by the parameter to minimize, a ★ denotes the best method, a + denotes a method with performances close to those of the best one and a − is for methods that show lower performances. Techniques introduced in [**cagli2015enhancing**] are highlighted by a grey background.

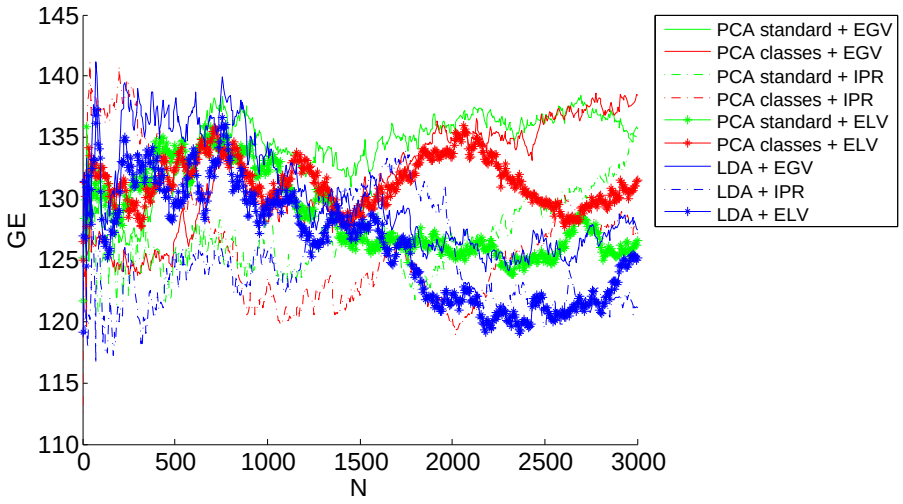


FIGURE 4.11: Degradation of linear-reduction-based template attacks in presence of misalignment.

Chapter 5

Kernel Discriminant Analysis

We tackle the dimensionality reduction problem in the context of profiling attacks against implementations protected by masking countermeasure. For such attacks, the attacker might have or not access to the random values drawn at every execution and used to mask the sensitive variables. If he have such a knowledge, then the dimensionality reduction problem turns to be equivalent to the case of unprotected implementations. Thus ,the classic statistics for the PoIs research (and we will concentrate over the SNR, see Sec. 2.2.4) and the linear dimensionality reduction techniques described in the previous chapter are still applicable and efficient. On the contrary, when such a knowledge is denied, linear techniques are completely inefficient: a non-linear function of the PoIs must considered in order to construct discriminant features from side-channel observations. In this chapter we propose to make use of Kernel Discriminant Analysis (KDA) technique to construct such a non-linear processing. To this aim we revisit the contents and the experimental results of the paper presented at CARDIS 2016 international conference [CDP16b]. After such a publication, the KDA has been compared to other non-linear dimensionality reduction techniques in [Ou+17], where manifold learning solutions such as ISOMAP, Locally Linear Embedding (LLE) and Laplacian Eigenmaps (LE) are proposed. Moreover, a use of the KDA in an unsupervised way to perform a higher-order SCA (as a key candidate distinguisher and not as a dimensionality reduction technique) has been proposed at CARDIS 2017 [Zho+17].

5.1 Motivation

When a masking countermeasure is properly applied, it ensures that every sensitive variable Z is randomly split into multiple shares $M_1, M_2, \ldots, M_d$ in such a way that a relation $Z = M_1 \star \cdots \star M_d$ holds for a group operation $\star$ (*e.g.* the exclusive or for the Boolean masking). The value d plays the role of a security parameter and the method is usually referred to as $(d-1)$th-order masking (since it involves $d-1$ random values). In many cases, especially for software implementations, the shares are manipulated at different times, and no time sample therefore shows dependency on Z: in order to recover such information the attacker is obliged to join information held by each of the d shares, executing a so-called dth-order SCA. In the great majority of the published higher-order attacks, the PoI selection during the pre-attack characterization phase is either put aside or made under the hypothesis that the random shares are known. Actually, the latter knowledge brings the problem back to the unprotected case. Here we relax this hypothesis and we consider situations where the values of the random shares are unknown to the adversary. We however assume that the adversary can characterize the leakage before attacking the implementation, by controlling the value of the target variable Z. These two assumptions

put our study in the context of *profiling attacks without knowledge of the masks*.

5.1.1 Getting information from masked implementations

The SNR estimation defined by (2.1) is an instrument to measure, point by point, the information held by the first-order moment of the acquisition, *i.e.* the information obtainable observing the variation of the mean of the acquisitions. We refer to such information as a *1st-order information*. In masked implementations, such information is null: in any time sample the mean is independent from Z due to the randomization provided by the shares, namely the quantity $\mathbb{E}[\vec{X}|Z = z]$, seen as a function of z, is constant, which implies that the SNR is asymptotically null over the whole trace.

When a $(d-1)$th-order masking is applied, the information about the shared sensitive target Z lies in some dth-order statistical moments of the acquisition,[1] meaning that for some d-tuples of samples $(t_1, \ldots, t_d)$ the quantity $\mathbb{E}[\vec{X}[t_1]\vec{X}[t_2]\cdots\vec{X}[t_d]|Z = z]$ (based on a dth-order raw moment) is not constant as a function of z (equivalently, $\mathbb{E}[(\vec{X}[t_1] - \mathbb{E}[\vec{X}[t_1]])\cdots(\vec{X}[t_d] - \mathbb{E}[\vec{X}[t_d]])|Z = z]$ is not constant, using the central moment). We can refer to the information obtainable observing such variation as a *dth-order information*. In order to let the SNR reveal it, and consequently let such information be directly exploitable by common attacks, the attacker must pre-process the traces through an extractor ϵ that renders the mean of the extracted data dependent on Z, *i.e.* such that $\mathbb{E}[\epsilon(\vec{X})|Z = z])$ is not constant as a function of z. In this way, the dth-order information is brought back to a 1st-order one.

Property 1 (SCA efficiency necessary condition). Let us assume that Z is represented by a tuple of shares M_i manipulated at d different times. Denoting $t_1, \ldots, t_d$ the time samples[2] where each share is handled, the output of an effective extractor needs to have at least one coordinate whose polynomial representation over the variables given by the coordinates of $\vec{X}$ contains at least one term divisible by the the dth-degree monomial $\prod_{i=1,\ldots,d}\vec{X}[t_i]$ (see *e.g.* [Car+14] for more information).

Remark 5.1. The use of central moments has been experimentally shown to reveal more information than the use of the raw ones [Cha+99; PRB09]. Thus we will from now on suppose that the acquisitions have previously been normalized, so that $\hat{\mathbb{E}}(\vec{x}_i) = \mathbf{0}$ and $\hat{\mathrm{Var}}(\vec{x}_i) = \mathbf{1}$. In this way a centred product coincides with a non-centred one.

We motivate hereafter through a simplified but didactic example, the need for a computationally efficient dimensionality reduction technique as preliminary step to perform an higher-order attack.

5.1.2 Some strategies to perform higher-order attack

We consider here an SCA targeting an 8-bit sensitive variable Z which has been priorly split into d shares and we assume that a reverse engineering and signal processing have priorly been executed to isolate the manipulation of each share in a region of ℓ time samples. This implies that our SCA now amounts to extract a Z-dependent information from leakage measurements whose size has been reduced to $d \times \ell$ time

[1] whence the name *dth-order attacks*

[2] not necessary distinct

samples. To extract such information three main approaches were proposed in literature until 2016.

The first one consists in considering d time samples at a time, one per region, and in testing if they jointly contain information about Z (*e.g.* by estimating the mutual information [RGV12] or by processing a Correlation Power Attack (CPA) using their centred product [Cha+99], etc.). Computationally speaking, this approach requires to evaluate ℓ^d d-tuples (*e.g.* 6.25 million d-tuples for $d = 4$ and $\ell = 50$), thus its complexity grows exponentially with d.

The second approach, that avoids the exhaustive enumeration of the d-tuples of time samples, consists in estimating the conditional pdf of the whole region: to this scope, a Gaussian mixture model is proposed in literature [LRP07; Lom+14] and the parameters of such a Gaussian mixture can be estimated through the expectation-maximization (EM) procedure. In [LRP07] 4 variants of the procedure are proposed according to a trade-off between the efficiency and the accuracy of the estimations; the most rough leads to the estimation of $256^{(d-1)}(\ell d)$ parameters (*e.g.* ≈ 3.4 billion parameters for $d = 4$ and $\ell = 50$), while the finest one requires the estimation of $256^{(d-1)}(1 + \frac{3 \cdot \ell d}{2} + \frac{(\ell d)^2}{2} - 1)$ parameters (*e.g.* ≈ 87 trillion parameters). Once again, the complexity of the approach grows exponentially with the order d.

The third approach, whose complexity does not increase exponentially with d, is the application of the higher-order version of the PP tool for the PoI selection, for which we give an outline hereafter. As will be discussed in Sec. 5.4.5, its heuristic nature is the counterpart of the relatively restrained complexity of this tool.

5.1.2.1 Higher-Order Version of Projection Pursuits

The dth-order version of PP makes use of the so-called *Moment against Moment Profiled Correlation* (MMPC) as objective function. The extractor ϵ^{PP} has the following form:

$$\epsilon^{PP}(\vec{x}) = (\vec{\alpha} \cdot \vec{x})^d \,, \tag{5.1}$$

where $\vec{\alpha}$ is a sparse projecting vector with d non-overlapping windows of coordinates set to 1, in correspondence with the identified points of interest. Actually, as will be discussed in Sec. 5.4.5, authors of [Dur+15] propose to exploit $\vec{\alpha}$ as a pointer of PoIs, but do not encourage the use of ϵ^{PP} as an attack preprocessing. The procedure is divided into two parts: a global research called *Find Solution* and a local optimization called *Improve Solution*. At each step of *Find Solution*, d windows are randomly selected to form a primordial $\vec{\alpha}$, thus a primordial ϵ^{PP}. A part of the training traces are then processed via ϵ^{PP} and used to estimate the dth-order statistical moments $\vec{m}_z^d = \hat{\mathbb{E}}_i[(\epsilon^{PP}(\vec{x}_i))_{i:z_i=z}])$, for each value of z. Then the Pearson correlation coefficient $\hat{\rho}$ between such estimates and the same estimates issued from a second part of the training set is computed. If $\hat{\rho}$ is higher than some threshold T_{det}, the windows forming $\vec{\alpha}$ are considered interesting[3] and *Improve Solution* optimises their positions and lengths, via small local movements. Otherwise, the $\vec{\alpha}$ is discarded and another d-tuple of random windows is selected.

[3] A further validation is performed over such windows, using other two training sets to estimate $\hat{\rho}$, in order to reduce the risk of false positives.

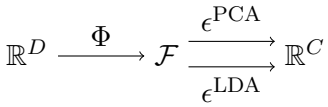


FIGURE 5.1: Performing LDA and PCA over a high-dimensional feature space.

The threshold T_{det} plays a fundamental role in this crucial part of the algorithm: it has to be small enough to promote interesting windows (avoiding false negatives) and high enough to reject uninteresting ones (avoiding false positives). A hypothesis test is used to choose a value for T_{det} in such a way that the probability of $\hat{\rho}$ being higher than T_{det} given that no interesting windows are selected is lower than a chosen significance level β.[4]

5.1.3 Foreword of this study

The exploitation of KDA technique in the way we propose in this chapter aims to exploit interesting d-tuples of time samples as the first strategy described in Sec. 5.1.2. It however improves it in several aspects. In particular, its complexity does not increase exponentially with d. Moreover, it may be remarked that such a first approach allows the attacker to extract interesting d-tuples of points, but does not provide any hint to conveniently combine them. This is an important limitation since finding a convenient way to combine time samples would raise the SCA efficiency, as already experimentally shown in [Bru+14], for $d = 1, 2$. Nevertheless in the SCA literature no specific method has been proposed for the general case $d > 2$. This study aims to propose a new answer to this question, while showing that it compares favourably to the PP approach.

5.2 Feature Space, Kernel Function and Kernel Trick

As described in Sec. 5.1.1, the hard part of the construction of an effective extractor is the detection of d time samples $t_1, \ldots, t_d$ where the shares leak. A naive solution, depicted in Fig. 5.1, consists in applying to the traces the centred product preprocessing for each d-tuple of time samples. Formally it means immerse the observed data in a higher-dimensional space, via a non-linear function

$$\Phi \colon \mathbb{R}^D \to \mathcal{F} = \mathbb{R}^{\binom{D+d-1}{d}} . \tag{5.2}$$

Using the Machine Learning language the higher-dimensional space $\mathcal{F}$ will be called *feature space*, because in such a space the attacker finds the features that discriminate different classes. Procedures involving a feature space defined as in (5.2) imply the construction, the storage and the management of $\binom{D+d-1}{d}$-sized traces; such a combinatorially explosion of the size of $\mathcal{F}$ is undoubtedly an obstacle from a computational standpoint.

[4] Interestingly, the threshold T_{det} depends on size of $\mathcal{Z}$ and not on the size of the training sets of traces. This fact disables the classic strategy that consists in enlarging the sample, making T_{det} lower, in order to raise the statistical power of the test (*i.e.* $\text{Prob}[\hat{\rho} > T_{det} | \rho = 1]$). Some developments of this algorithm have been proposed [DS15], also including the substitution of the MMPC objective function with a *Moments against Samples* one, that would let T_{det} decrease when increasing the size of the training set.

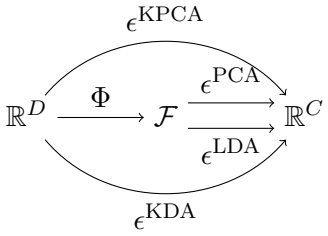


FIGURE 5.2: Applying KDA and KPCA permits to by-pass computations in $\mathcal{F}$.

In Machine Learning a stratagem known as *kernel trick* is available for some linear classifiers, such as Support Vector Machine (SVM), PCA and LDA, to turn them into non-linear classifiers, providing an efficient way to implicitly compute them into a high-dimensional feature space. This section gives an intuition about how the kernel trick acts. It explains how it can be combined with the LDA, leading to the so-called KDA algorithm, that enables an attacker to construct some non-linear extractors that concentrate in few points the d-th order information held by the side-channel traces, without requiring computations into a high-dimensional feature space, see Fig. 5.2.

The central tool of a kernel trick is the *kernel function* $K\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$, that has to satisfy the following property, in relation with the function Φ:

$$K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)\,, \tag{5.3}$$

for each $i, j = 1, \ldots, N$, where $\cdot$ denote the dot product.

Every map Φ has an associated kernel function given by (5.3), for a given set of data. The converse is not true: all and only the functions $K\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ that satisfy a convergence condition known as *Mercer's condition* are associated to some map $\Phi : \mathbb{R}^D \to \mathbb{R}^S$, for some S. Importantly, a kernel function is interesting only if it is computable directly from the rough data $\vec{x}$, without evaluating the function Φ.

The notion of kernel function is illustrated in the following example.

Example 1. Let $D = 2$. Consider the function

$$\begin{aligned} &K\colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R} \\ &K\colon (\vec{x}_i, \vec{x}_j) \mapsto (\vec{x}_i \cdot \vec{x}_j)^2\,, \end{aligned} \tag{5.4}$$

After defining $\vec{x}_i = [a, b]$ and $\vec{x}_j = [c, d]$, we get the following development of K:

$$K(\vec{x}_i, \vec{x}_j) = (ac + bd)^2 = a^2c^2 + 2abcd + b^2d^2\,, \tag{5.5}$$

which is associated to the following map from $\mathbb{R}^2$ to $\mathcal{F} \subset \mathbb{R}^3$:

$$\Phi(u, v) = [u^2, \sqrt{2}uv, v^2] \tag{5.6}$$

Indeed $\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j) = a^2c^2 + 2abcd + b^2d^2 = K(\vec{x}_i, \vec{x}_j)$. This means that to compute the dot product between some data mapped into the 3-dimensional space $\mathcal{F}$ there is no need to apply Φ: applying K over the 2-dimensional space is equivalent. This trick allows the short-cut depicted in Fig. 5.2.

In view of the necessary condition exhibited by Property 1, the function $K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j)^d$, hereafter named *dth-degree polynomial kernel function*, is the convenient

choice for an attack against implementations protected with $(d-1)$th-order masking. It corresponds to a function Φ that brings the input coordinates into a feature space $\mathcal{F}$ containing all possible d-degree monomials in the coordinates of $\vec{x}$, up to constants. This is, up to constants, exactly the Φ function of (5.2).[5]

5.3 Kernel Discriminant Analysis

The equation (5.3) shows that a kernel function K allows to compute the dot product between elements mapped into the feature space $\mathcal{F}$ (5.3). By extension, any procedure that only implies the computation of dot products between elements of $\mathcal{F}$, can be executed exploiting a kernel function. Starting from this remark, the authors of [SSM98; SM99] have shown that the PCA and LDA procedures can be adapted to satisfy the latter condition, which led them to define the KPCA and KDA algorithms. The latter one is described in Sec. 5.3.1. The interested reader will find the formal derivation of KPCA in Appendix C, reported as a way of example of how one can translate the classical PCA algorithm (and the class-oriented version) in such a way to never access data, but only dot product between data. The way to derive the KDA procedure reported below is analogous; the interested reader might refer to [SM99], or to [BA00] for the multi-class version.

5.3.1 KDA for dth-order masked side-channel traces

Given a set of labelled training side-channel traces $(\vec{x}_i, z_i)_{i=1,\dots,N}$ and the kernel function $K(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y})^d$:

1) Construct a matrix $\mathbf{M}$ (acting as *between-class scatter matrix*):

$$\mathbf{M} = \sum_{z \in \mathcal{Z}} N_z (\vec{M}_z - \vec{M}_T)(\vec{M}_z - \vec{M}_T)^\intercal \,, \tag{5.7}$$

where $\vec{M}_z$ and $\vec{M}_T$ are two N-sized column vectors whose entries are given by:

$$\vec{M}_z[j] = \frac{1}{N_z} \sum_{i:z_i=z}^{N_z} K(\vec{x}_j, \vec{x}_i) \tag{5.8}$$

$$\vec{M}_T[j] = \frac{1}{N} \sum_{i=1}^{N} K(\vec{x}_j, \vec{x}_i) \,. \tag{5.9}$$

2) Construct a matrix $\mathbf{N}$ (acting as *within-class scatter matrix*):

$$\mathbf{N} = \sum_{z \in \mathcal{Z}} \mathbf{K}_z (\mathbf{I} - \mathbf{I}_{N_z}) \mathbf{K}_z^\intercal \,, \tag{5.10}$$

[5] Other polynomial kernel functions may be more adapted if the acquisitions are not free from d'th-order leakages, with $d' < d$. Among non-polynomial kernel functions, we effectuated some experimental trials with the most common Radial Basis Function (RBF), obtaining no interesting results. This might be caused by the infinite-dimensional size of the underlying feature space, that makes the discriminant components estimation less efficient.

where $\mathbf{I}$ is a $N_z \times N_z$ identity matrix, $\mathbf{I}_{N_z}$ is a $N_z \times N_z$ matrix with all entries equal to $\frac{1}{N_z}$ and $\mathbf{K}_z$ is the $N \times N_z$ sub-matrix of $\mathbf{K} = (K(\vec{x}_i, \vec{x}_j))_{\substack{i=1,\dots,N \\ j=1,\dots,N}}$ storing only columns indexed by the indices i such that $z_i = z$.

3) Regularize the matrix $\mathbf{N}$ for computational stability:

$$\mathbf{N} = \mathbf{N} + \mu\mathbf{I} \quad \text{see 5.4.2;} \tag{5.11}$$

4) Find the non-zero eigenvalues $\lambda_1, \dots, \lambda_Q$ and the corresponding eigenvectors $\vec{\nu}_1, \dots, \vec{\nu}_Q$ of $\mathbf{N}^{-1}\mathbf{M}$;

5) Finally, the projection of a new trace $\vec{x}$ over the ℓ-th non-linear dth-order discriminant component can be computed as:

$$\epsilon_\ell^{\mathrm{KDA}}(\vec{x}) = \sum_{i=1}^{N} \boldsymbol{\nu}_\ell[i] K(\vec{x}i, \vec{x}) \,. \tag{5.12}$$

For the reasons discussed in Sec. 5.2, the right-hand side of (5.12) may be viewed as an efficient way to process the ℓth coordinate of the vector $\epsilon^{LDA}(\Phi(\vec{x}))[\ell] = \vec{w}_\ell \cdot \Phi(\vec{x})$, without evaluating $\Phi(\vec{x})$. The entries of $\vec{w}_\ell$ are never computed, and will thus be referred to as *implicit coefficients* (see Sec. 5.3.2 below). It may be observed that each discriminant component $\epsilon_\ell^{\mathrm{KDA}}(\cdot)$ depends on the training set $(\vec{x}_i, z_i)_{i=1,\dots,N}$, on the kernel function K and on the regularization parameters μ.

5.3.2 The implicit coefficients

As already said, when the dth-degree polynomial kernel function is chosen as kernel function, the KDA operates implicitly in the feature space of all products of d-tuples of time samples. In order to investigate the effect of projecting a new trace $\vec{x}$ over a component $\epsilon_\ell^{\mathrm{KDA}}(\vec{x})$, we can compute for a small d the implicit coefficients that are assigned to the d-tuples of time samples through (5.12). For $d = 2$ we obtain that in such a feature space the projection is given by the linear combination computed via the coefficients shown below:

$$\epsilon_\ell^{\mathrm{KDA}}(\vec{x}) = \sum_{j=1}^{D}\sum_{k=1}^{D} [(\vec{x}[j]\vec{x}[k]) \underbrace{(\sum_{i=1}^{N} \boldsymbol{\nu}_\ell[i]\vec{x}_i[j]\vec{x}_i[k])}_{\text{implicit coefficients}}] \tag{5.13}$$

5.3.3 Computational complexity analysis

The order d of the attack does not significantly influence the complexity of the KDA algorithm. Let N be the size of the training trace set and let D be the trace length, then the KDA requires:

- $\frac{N^2}{2}D$ multiplications, $\frac{N^2}{2}(D-1)$ additions and $\frac{N^2}{2}D$ raising to the d-th power, to process the kernel function over all pairs of training traces
- $(D + C)$ multiplications, $(D + C - 2)$ additions and 1 raising to the d-th power for the projection of each new trace over C KDA components,
- the cost of the eigenvalue problem, that is $O(N^3)$.

In next sections we discuss the practical problems an attacker has to deal with when applying the KDA. The argumentation is conducted on the basis of experimental results whose setup is described hereafter.

5.4 Experiments over Atmega328P

5.4.1 Experimental Setup

The target device is an 8-bit AVR microprocessor Atmega328P and we acquired power-consumption traces thanks to the ChipWhisperer platform [OC14].[6] From the acquisitions we extracted some traces composed of 200 time samples, corresponding to 4 clock cycles (see Fig.5.7(a) or 5.7(b) upper parts). Depending on the attack implementation, we ensure that the acquisitions contain either 2,3 or 4 shares respectively for $d = 2, 3$ or 4. The shares satisfy $M_1 \oplus \cdots \oplus M_d = Z$, where Z takes values in $\mathcal{Z} = \mathbb{F}_2^8$ and represents one byte of the output of the first rount SubByte operation in AES: $Z = \mathrm{Sbox}(P \oplus K^\star)$. The goal of the attack is to recover the subkey $K^\star$. The plaintext P is assumed to be known and the M_i are assumed to be unknown random uniform values. The profiling phase is divided in two sub-phases that exploit two distinct datasets. The first sub-phase, that we will refer to as KDA training phase, aims at constructing the dimensionality reduction function by means of KDA algorithm. It exploits a KDA dataset $\mathcal{D}_{\text{train}} = (\vec{\mathcal{X}}_{\text{train}}, \mathcal{Y}_{\text{train}})$ of size $N = 8960$. A known fixed subkey is used to acquire such a dataset, the plaintexts have been chosen to balance the number of classes ($N_z = \frac{8960}{256} = 35$ for each $z \in \mathcal{Z}$). We fixed the dimension C at the value 2 (except for the 2-class KDA for which we chose $C = 1$, see Remark 5.3): we therefore tried to build extractors $\epsilon^{\text{KDA}}(\cdot) = (\epsilon_1^{\text{KDA}}(\cdot), \epsilon_2^{\text{KDA}}(\cdot))$ mapping traces of size 200 samples into new traces composed of 2 coordinates.[7] Afterwards, a bivariate Gaussian TA (see Sec. 2.2.3.1) is run. Such an attack demands for a proper profiling phase, consisting in the estimation of the class-conditional probabilities. It justifies the second profiling sub-phase anticipated above. It will be referred to as profiling phase and will be performed exploiting a distinct profiling dataset $\mathcal{D}_{\text{profiling}} = (\vec{\mathcal{X}}_{\text{profiling}}, \mathcal{Y}_{\text{profiling}})$, collecting 1000 traces per class, under a fixed known key. The choice of not reusing $\mathcal{D}_{\text{train}}$ as profiling dataset for the Gaussian TA has been done in order to reduce the overfitting risk, discussed in general in Sec. 3.1.4 and that will be discussed in the particular case of KDA in Sec. 5.4.2. The extractor ϵ^{KDA}, has a different behaviour when applied to the traces used to train it and to some new traces: this is inevitable since ϵ^{KDA} uses all training traces has its proper parameters (5.12). Thus, a new unobserved profiling set is mandatory in order to obtain an uncorrupted profiling.

As discussed in Remark 5.1, the KDA training traces are normalized. The average trace and the standard deviation trace used to perform the normalization are stored and reused to center the profiling and attack traces before projecting them onto the KDA components. In this way the profiling and attack traces present a form as similar as possible to the training ones.

[6] This choice has been done to allow for reproducibility of the experiments.

[7] As we have seen in Chapter 4, for PCA and LDA methods a good component selection is fundamental to obtain an efficient subspace, and that the first components not always represent the best choice. This is likely to be the case for the KDA as well, but in our experiments the choice of the two first components $\epsilon_1^{\text{KDA}}, \epsilon_2^{\text{KDA}}$ turns out to be satisfying, and therefore to simplify our study we preferred to not investigate other choices.

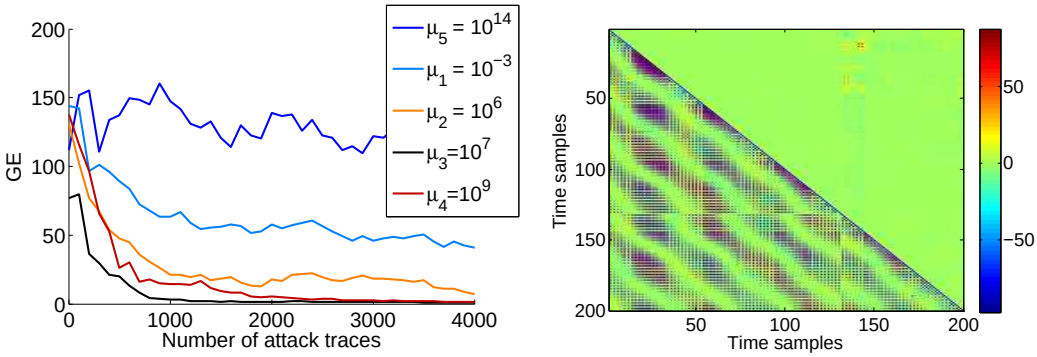


FIGURE 5.3: On the left: template attack guessing entropy vs the number of traces for the attack phase, varying for choices of the constant μ (5.11). On the right: the implicit coefficients assigned to pairs of time samples for μ_3 (upper triangular part) and μ_5 (lower triangular part).

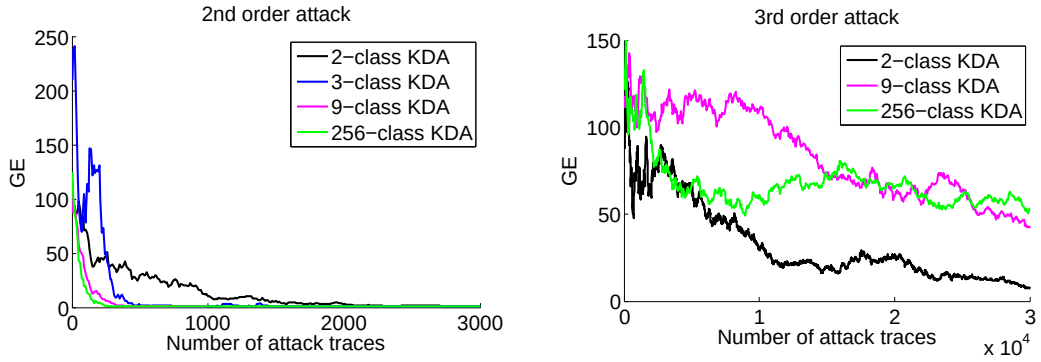
5.4.2 The Regularization Problem

By construction the matrix $\mathbf{N}$ in (5.10) is not positive-definite, which is one of the reasons why in [SM99], where the application of a kernel trick to LDA is proposed for the first time, the authors propose the regularization (5.11) recalled hereafter:

$$\mathbf{N} = \mathbf{N} + \mu\mathbf{I}\,. \tag{5.14}$$

When applying such a regularization, the choice of the constant μ is crucial. Beyond the form of the kernel function, μ is the unique hyper-parameter of the model constructed by the KDA algorithm, in the sense explained in Sec. 3.1.5. Its value cannot be learned from data and has to be priorly fixed somehow. For sure it has to be large enough to ensure that $\mathbf{N}$ turns to a positive-definite matrix, but we experimentally observed that the minimal μ for which the positive-definitiveness of $\mathbf{N}$ is attained is often far from being the one that provides a good extractor. In Fig. 5.3 (left) we observe the efficiency of a template attack run in combination with a KDA extractor. The matrix $\mathbf{N}$ is positive-definite for $\mu_1 = 10^{-3}$ but the value that provides the best extractor is much higher (namely $\mu_3 = 10^7$). Still raising the value of μ degrades the quality of the extractor. The right part of Fig. 5.3 shows the implicit coefficients of the extractor (see (5.13)) obtained under μ_3 (upper triangular part) and under μ_5 (lower triangular part). The extractor corresponding to the former one leads to a successful attack and has high values concentrated over the interesting windows; the extractor corresponding to the latter one leads to an unsuccessful attack and shows lack of localization around interesting parts of the trace, highlighting the fact that the KDA tool failed in detecting generalisable class-distinguishing features in this case.

The regularization (5.11) is a proper regularization in the sense discussed in Sec. 3.1.4: it is not only a way to make the problem computationally stable (which explains why the minimal μ making $\mathbf{N}$ positive-definite may not be a good choice), but also a answer to the overfitting phenomenon. In the case of the KDA the overfitting is observable when ϵ^{KDA} almost perfectly separates the training traces in their classes, while failing in separating the profiling and the attack ones. In [SM99] it is shown that the regularization (5.11) corresponds to the additional requirement for $\vec{\nu}$ to have a small norm $\|\vec{\nu}\|^2$. As every regularization technique, it makes the method



(a) (b)

FIGURE 5.4: Comparison between 2-class,3-class, 9-class and 256-class KDA in 2nd-order context (a) and in 3rd-order context (b). For 2nd-order the KDA is efficient in providing separability between 256 classes, allowing an optimal attack. In 3rd-order context the training data are not enough to succeed the 256-class learning phase. Decreasing the number of classes to be distinguished raises the efficiency of the learning problem and thus of the attack.

less accurate in the learning phase, but in some cases more likely to correctly operate on new examples.

Remark 5.2. Another regularization strategy may be to search for sparse vectors of implicit coefficients (see (5.13)). This alternative might be more suitable for the side-channel context, since it would promote localized solutions, *i.e.* projections for which only a few d-tuples of time samples contribute to the construction of the extractor (see Assumption 1 in Chapter 4 for an analogy in $1st$-order context). This approach is left for future developments.

Some heuristics exist to choose the constant μ, *e.g.* the average of the diagonal entries [LZO06] or the minimal constant that let $\mathbf{N}$ be diagonally dominant (implying positive-definite). In [CL06] Centeno *et al.* propose a maximization strategy to find the optimal regularization parameter, based on a probabilistic approach. We did not apply such heuristics for our study, but we consider them in order to fix a grid of values for μ to be tested. Then, as explained in Sec. 3.1.5 we chose an approach based over a validation, in order to fix the final value of μ before performing the attack phase. To perform such validation we chose the SNR as performance measure for the extractor provided by the KDA and $\mathcal{D}_{\text{profiling}}$ as validation dataset.

5.4.3 The Multi-Class Trade-Off

As discussed in Sec. 4.3, the LDA, and by consequence the KDA, looks for a subspace of the feature space to optimally separate some given classes. The performance of the KDA algorithm raises with the size N of the training set. Nevertheless, the number of examples might be bounded by the acquisition context, and even when the N can be very high, it may be interesting to minimize it since the KDA complexity is $O(N^3)$. A trade-off must therefore be found between accuracy and efficiency. Assuming that the size of the training set is fixed to N, which controls the efficiency, a way to gain in accuracy may be found by appropriately adjusting the number of classes to distinguish: intuitively, the more examples per class, the more accurate the detection of a separating subspace. Then, if the total number of training traces

is fixed, in order to raise the number of traces per class, a smaller number of classes must be considered. To do so, a non-injective model $m(\cdot)$ can be introduced, to create a smaller set of labels $m(\mathcal{Z})$ from the initial set $\mathcal{Z}$. The reduced number of classes, *i.e.* the number of labels assigned to the training set after applying the model m, will be denoted by W (it is the cardinality of $m(\mathcal{Z})$). As discussed in Sections 2.2.1.2 and 2.2.1.3, a widely-accepted power-consumption model for side-channel traces is provided by the Hamming Weight (HW) function, thus we consider and experimentally compare the following models:

- 2-class model ($W = 2$)

$$\begin{cases} m(z) = 0 \text{ if } \mathrm{HW}(z) < 4 \\ m(z) = 1 \text{ if } \mathrm{HW}(z) \geq 4 \end{cases}$$

- 3-class model ($W = 3$)

$$\begin{cases} m(z) = 0 \text{ if } \mathrm{HW}(z) < 4 \\ m(z) = 1 \text{ if } \mathrm{HW}(z) = 4 \\ m(z) = 2 \text{ if } \mathrm{HW}(z) > 4 \end{cases}$$

- 9-class model ($W = 9$)

$$m(z) = \mathrm{HW}(z) \, .$$

Remark 5.3. The separating subspace given by the KDA has maximal dimension $(W - 1)$, *i.e.* $Q \leq (W - 1)$ in point 4 of Sec. 5.3.1. When $W = 2$ a single discriminant component $\epsilon_1^{\mathrm{KDA}}$ is available. In this case we cannot run a bivariate template attack as we do with other extractors, thus we run a univariate one.

A balanced training set of size $N = 9000$ (instead of 8960) has been used to run the experiments for 2-class, 3-class and 9-class KDA. For the sake of consistency[8] between the pre-processing phase and the attack phase, when a non-injective model is applied to the labels of the training set to reduce the number of classes, the same model is exploited to run the template attack: W templates (one per each class) are estimated from the profiling set and compared to the attack traces. Thus, results of the experimental comparison of these different multi-class approaches depicted in Fig. 5.4 are obtained using different template attacks. It may be remarked that as W decreases the efficiency of the attack is supposed to decrease as well, because each attack trace contributes in distinguishing the right key $K^\star$ only from a growing-size set of indistinguishable hypotheses.

In 2nd-order context, it can be observed in Fig. 5.4 that the KDA is provided with sufficient training traces to succeed a 256-class separation, which allows the finest characterization of the leakage, and leads as expected (see Remark **??**), to the most efficient template attack. Moving to the 3-rd order context, the available training set is insufficient to make the multi-class approach succeed; nevertheless, turning the problem into a 2-class problem turns out to be a good strategy to trade extraction accuracy for attack efficiency.

An idea to avoid an excessive reduction of the number of separable classes W is given in the machine learning literature: it consists in treating the W-class problem as multiple 2-class problems. Two different *modus operandi* exist: the *one-vs-one* and

[8] A different approach is analysed in Sec. 5.4.4.

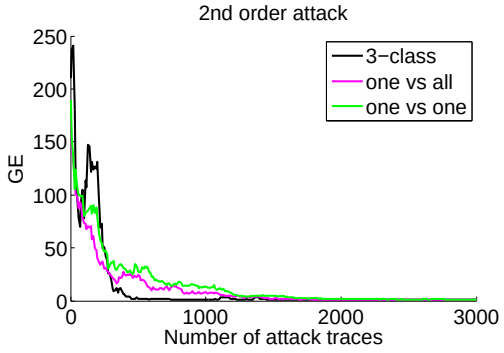


FIGURE 5.5: Performance of template attacks run over 3-class KDA subspaces: multi-class, one vs one and one vs all approaches compared.

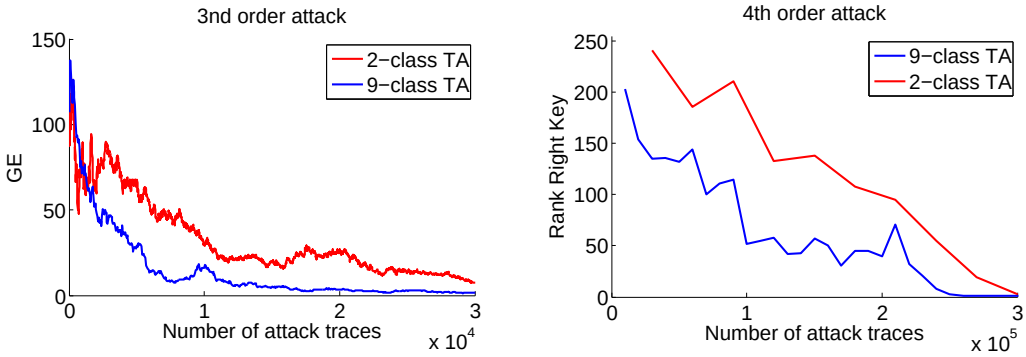


FIGURE 5.6: Left: guessing entropy (over 10 independent tests) for a 2-class and a 9-class 3rd-order template attack. Right: right key rank of a 2-class and a 9-class 4th-order template attack.

the *one-vs-all*. When applied to our context, the one-vs-one approach determines for each pair of classes the 1-dimensional subspace that best separates them and exploits all the obtained subspaces to run an attack (for W classes we obtain $\binom{W}{2}$ dimensions and we run a $\binom{W}{2}$-variate template attack). The one-vs-all approach looks for dimensions that best separate each class from all the others (obtaining W projections in total).

We tested this approach in the 3-class case: in this way the one-vs-one and the one-vs-all approaches provide both 3 dimensions that we use to run a 3-variate template attack, and that we compare to the 3-class multi-class approach with bivariate template attack. Our experimental results, summed up in Fig. 5.5, show that no gain is obtained by the 2-classes strategies.[9] We therefore chose to not consider them for the higher-order experiments.

5.4.4 Asymmetric Preprocessing/Attack Approach

In previous section we appealed a consistency principle to justify the choice of running a W-class template attack after a W-class KDA extraction. Here we propose a different reasoning: the consistency principle does not grant that an extractor ϵ^{KDA} trained with W classes is not able to separate W' classes with $W' > W$. As seen

[9] We think that is result is quite data-dependant, so the use of such an approach is not discouraged in general.

in Sec. 5.3.2, an extractor ϵ^{KDA} always implicitly performs a weighed sum, via the implicit coefficients, of centred products of time samples. If ϵ^{KDA} is effective, the implicit coefficients which have the highest magnitude must correspond to time samples corresponding to the manipulation of sensitive data (*e.g.* the variable shares when masking is applied). This property is not necessarily related to the number of classes used to train the extractor.

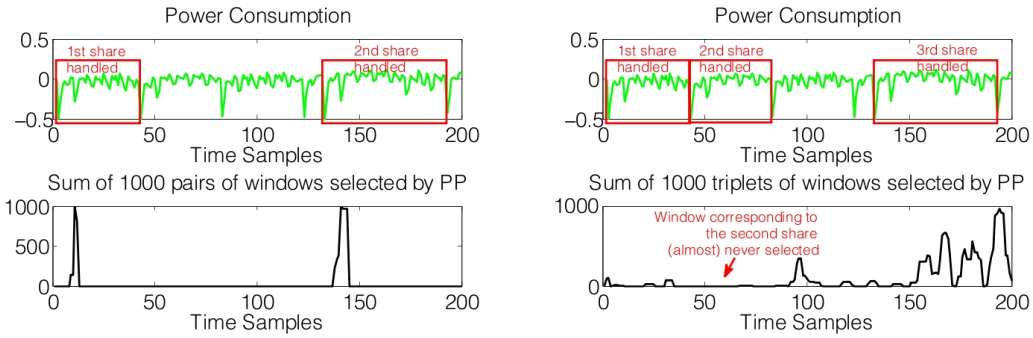
Based on the reasoning above, we experienced the 3rd-order and the 4th-order attacks in an asymmetric way: as preprocessing we performed a 2-class KDA, which gave best performances compared to others in the 3rd-order context (Fig. 5.4(b)), then we performed a 9-class template attack, in order to raise the accuracy of the profiling and the efficiency of the attack. The results are depicted in Fig. 5.6 and confirm that, for our experimental data, this approach is sound: in both cases, using the same extractor trained with 2 classes and the same attack traces, the 9-class approach outperforms the 2-class one.

5.4.5 Comparison with Projection Pursuits

To get a fair comparison, we run the PP algorithm (see Sec. 5.1.2.1) over the same training set used to evaluate the KDA in Sec.5.4. The best results in the 2nd-order context were obtained with the HW model (*i.e.* $|\mathcal{Z}| = 9$). In this case T_{det} is fixed to 0.7. Since 4 training sets are required, the 9000 training traces are split in 4 equally-sized groups. Experimental observations allowed to fix $W_{len} = 5$, consequently suggesting $minWS = 1$, $maxWS = 15$ and consistent global and local movements and resizes. Given the heuristic asset of the algorithm, we run it 1000 times for $d = 2$ and for $d = 3$. An overview of the global behaviour of the obtained results is depicted in Figs 5.7(a) and 5.7(b): the lower parts of the figures show the sum of the 1000 outputs of the algorithm. We recall that each coordinate of $\vec{\alpha}$ is set to 1 for the windows identified to be of interest, and to 0 elsewhere, so for each time sample the sum of the values (0 or 1) assigned by the 1000 attempts give an intuition about its likelihood to be considered as interesting by the PP method. It can be observed that in the 2-nd order case (Fig. 5.7(a)) the results are excellent: 100% of the tests highlight an informative part of the two clock-cycles where the sensitive shares are manipulated.[10] It means that $\epsilon^{PP}(\vec{X})$ always contains information about Z and a successful attack can be mounted over such extracted traces. The efficiency of such an attack depending on many factors, there is no interest in comparing it to the performances of the template attacks run in 2nd-order context using ϵ^{KDA} and depicted in Fig. 5.4(a). As it may be observed in Fig. 5.7(b), in the 3-rd order case the experimental results are completely different: almost no $\vec{\alpha}$ selects the clock-cycle where the second share is manipulated. Thus in this case the PP approach fails: $\epsilon^{PP}(\vec{X})$ does not contain information about Z, so any attack launched over the extracted traces would fail, while ϵ^{KDA} still allows successful attacks in 3rd-order and 4th-order case, as depicted in Fig. 5.6.

We conclude that the KDA approach is a valuable alternative to the PP one, especially in contexts where the training set size is bounded and independent from the order d of the attack.

[10] It can be observed that the regions selected by ϵ^{PP} correspond to those for which the ϵ^{KDA} exhibits the highest magnitude implicit coefficients (Fig. 5.3, upper-triangular part on the right)



(a) (b)

FIGURE 5.7: (a) Overview of PP outputs in 2nd-order context. (b) Overview of PP outputs in 3rd-order context.

5.5 Conclusions and Drawbacks

In this chapter we analysed the use of the KDA method to extract small-sized informative features from side-channel acquisitions protected by a $(d-1)$th-order masking countermeasure. The KDA naturally extends the LDA technique to the generic dth-order context. It requires the choice of a so-called kernel function. We proposed to choose a polynomial kernel function, because it perfectly fit the necessary condition to perform a higher-order side-channel attack. Indeed, in this way the obtained extractor provides the linear combination of all possible dth-degree monomial in the time coordinates of the traces, which maximises the SNR. The main obstacle to the problems of PoI selection and dimensionality reduction in higher-order side-channel context is given by the fact the size of the space containing all possible dth-degree monomials explodes combinatorially while d grows. Nevertheless, the KDA only implicitly operates in such a space, by means of a so-called kernel trick, implying that its complexity is independent from the order d. This property represents the main advantage of the KDA. Experiments described in this chapter in 2nd-order, 3rd-order and 4th-order contexts confirmed that such an approach is effective. Anyway, this approach presents some drawbacks, discussed hereafter.

Regularization hyper-parameter First of all, to apply this methodology an attacker has to deal with choice of a regularization hyper-parameter. This problem still appears unsolved in subsequent studies [Zho+17].

Non scalability to big training set The computational cost of the optimization problem is affected by the number of side-channel traces it uses for the training. This obliges the attacker to find a good trade-off between the efficiency of the information extraction, its accuracy and the efficiency of the underlying attack, through a careful choice of the target classification model. Besides the computational cost, the size of the training set also affects the memory complexity of the dimensionality reduction model: training traces cannot be forgot after the training of ϵ^{KDA} but have to be stored in memory. Bishop assigned to this characteristic with the adjective *memory-based* [Bis06, Chapter 6]. Indeed, observing the form of the KDA extractor (5.12), one can remark that each time sample of each training trace makes part of the parameters defining it, together with the entries of the eigenvectors $\vec{\nu}_1, \ldots, \vec{\nu}_Q$.

This might be a surprisingly huge number of parameters: for example in our experiments, the extractor $\epsilon^{\mathrm{KDA}} : \mathbb{R}^2 00 \rightarrow \mathbb{R}^2$ constructed exploiting a 8960-sized training set counts $(8960 + 2) \times 200 = 1.792.400$ parameters. In the case of 2nd-order context, this number is much higher than the number of implicit coefficients assigned to all possible 2nd-degree monomials in time samples, which is $\binom{200+2-1}{2} = 20100$.

Misalignment Affection The KDA being an efficient way to perform LDA in a larger feature space, it maintains the same weakness than the LDA to trace misalignment, discussed in Sec. 4.5.

Two-Phased Approach The approach presented in this chapter (and in Chapter 4 as well) is characterised by being two-phased. Indeed, a preliminary training has to be done in order to construct the extractor ϵ, that plays the role of preprocessing for side-channel traces. Then, such an extractor is applied to traces and a second profiling phase has to be performed in order to construct the generative model that characterises the Gaussian template attack. In the specific case of KDA, these two preliminary phases demand the exploitation of two different profiling set, as discussed in Sec. 5.4.1, which might be a great disadvantage in contexts where profiling acquisitions are bounded. Anyway, there is a general greater disadvantage of this two-phased approach, which is the fact that the preprocessing part, aiming in reducing the dimensionality of the samples, inevitably reduces the information held by side-channel traces, and such a pruning is mainly guided by some prior assumptions about the form informative parts of the data takes. For example, the fact that the polynomial kernel function proposed in this chapter fits with the necessary condition given in Property 1, does not guaranty that a linear combination of dth-degree monomials is the most efficient preprocessing to extract sensitive information from the traces. Even when such a linear combination is chosen to maximise a precise well-chosen criterion, in case of KDA it is chosen to maximise the SNR of projected data, through the Rayleigh quotient condition, this criterion does not directly coincide with the goal of the attack, *i.e.* construct a classifier that allows to optimally distinguish the right secret key of the attacked algorithm from the wrong ones, or at least that allow to optimally classify the sensitive variable value handled during the acquisition of the attack traces. This same drawback of dimensionality reduction techniques is present in any preprocessing strategy, *e.g.* in realignment techniques: a preprocessing aiming at realign data has a partial objective (the resynchronization) that does not coincide with the final goal of the attack, thus inject a risk of degrading data quality with respect to the final goal. This remark about data preprocessing is not specific to SCA context. Indeed, it is the one that pushed the Machine Learning community to develop the Deep Learning (DL) branch: as we will see in next chapter, and as anticipated in Sec. 1.3.3, DL models are conceived to integrate in a unique optimising process (the learning phase) any preprocessing with the model construction.

Chapter 6

Convolutional Neural Networks

In this chapter we explore a new strategy to perform profiling SCAs, addressing the misalignment issue and endorsing the Deep Learning (DL) paradigm. To this aim we revisit the publication [CDP17] appeared at CHES 2017.

6.1 Introduction

Often Machine Learning approach declines in multiple pre-processing phases such as data alignment, feature selections or dimensionality reduction, followed by a final model optimisation. This is the case for the SCAs routines that we considered in previous chapters: a dimensionality reduction pre-processing ϵ is previously learned an applied, followed by a Gaussian template parameters estimation that are used to construct the model which solves the side-channel classification task. Deep Learning is a branch of Machine Learning whose characteristic is to avoid any preliminary pre-processing step from the model construction work-flow. For example, in Deep Learning the data dimensionality reduction is not necessarily explicitly performed by a distinct learned function ϵ. Thus, DL models are in charge of directly and implicitly extracting interesting features and of estimating the opportune model to solve the task. The model is searched in a family of models that are composed by a cascade of parametrisable layers, which may be optimised in a single global learning process. Such models are called *Artificial Neural Network*, or simply *Neural Networks* (NNs).

By construction, NNs are the ML answer to the drawback of work-flows we analysed in previous chapters and discussed as *two-phased approach drawback* in last section of Chapter 5. Actually, NNs are answers to other drawbacks pointed out in the same section.
In particular they are not memory-based. It implies that, after the training phase whose computational complexity is influenced by the size of the training set, they do not need to access the training set any more. By consequence, the obtained model is in general faster in processing new data, with respect to those obtained *via* kernel machines: the form of the obtained model (*i.e.* the so-called *architecture*) is independent from the size of the training set. This property, together with other computational optimizations they allow that will be discussed later, make NNs more easily scalable for huge training sets.
Finally, we pointed out as drawback of techniques analysed in previous chapters their weakness to trace misalignment. In DL literature, a family of models called *Convolutional Neural Networks* (CNNs) has been developed to treat difficulties usually met in image processing as misalignments, scaling, rotations, ...We claim in this chapter, and verify through various experiments, that such CNNs provide an

attack strategy that is robust to misalignment countermeasure. Attacks proposed in this chapter are performed against non-masked implementation. Nevertheless, since NNs are in general non-linear models, they naturally well-fit also the higher-order attack context, as discussed in [MPP16] and [Pro+18].
The CNNs are a generalisation of the simplest NN architecture, named *Feed-forward Neural Networks* or *Multi-Layer Perceptron* (MLP). The name *feedforward* is due to the fact that for this kind of models information Ô¨Çows from the input to the output, through the intermediate computations, without any feedback connection in which outputs of the model are fed back into itself. This is in opposition to the so-called *Recurrent Neural Network* structures. A description of Neural Networks models, and classification-oriented MLP in particular, is provided in next section.

6.2 Neural Networks and Multi-Layer Perceptrons

As anticipated in Chapters 2 and 3, we are interested in the NNs' solutions for the classification task, as we remarked a strong analogy between profiling SCAs and classical ML classification task. We recall from Chapters 3 that for the classification task, the learning algorithm is asked to construct a function $F\colon \mathbb{R}^D \to \{0,1\}^{|\mathcal{Z}|}$, where elements of $\mathcal{Z}$, *i.e.* the set of classes, are here expressed *via* the *one-hot encoding*. The output of such a function is said to be *categorical*, *i.e.* $\mathcal{Z}$ is a discrete finite set. A variant of the classification task consists in finding a function $F\colon \mathbb{R}^D \to [0,1]^{|\mathcal{Z}|}$ defining a probability distribution over classes. Often for this task, NNs are exploited to create discriminative models, *i.e.* models that directly approximate the latter function F that describes the posterior conditional probability of a label given the observed trace. This is in opposition to the Template Attack we exploited in previous chapters, that is based over the construction of generative models, *i.e.* the approximation of the *templates*, which coincide with the conditional probabilities of the traces given a label, as described in Sec. 2.2.3.1.

Using NNs the function F is obtained by combining several simpler functions, called *layers*. An NN has an *input layer* (the identity over the input datum $\vec{x}$), an output layer (the last function) and all other layers are called *hidden* layers. The output of F is a $|\mathcal{Z}|$-sized vector $\vec{y}$ of scores for the $|\mathcal{Z}|$ labels. Such a vector might or not represent the approximation of a probability distribution. The nature of the NN's layers, their number and their dimension in particular, is called the *architecture* of the NN. All the parameters that define an architecture, together with some other parameters that govern the training phase, are its *hyper-parameters*. The so-called *neurons*, that give the name to the NNs, are the computational units of the network and essentially process a scalar product between the coordinates of its input and a vector of *trainable weights* (or simply *weights*) that have to be *trained*. Each layer processes some neurons and the outputs of the neuron evaluations will form new input vectors for the subsequent layer.

The *Multi-Layer Perceptrons* (MLPs) are a family of NN's architectures, associated with a function F that is composed of multiple linear functions and some non-linear functions, called *activations*.

We can express a typical classification-oriented MLP by the following equation:

$$F(\vec{x}) = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\vec{x}) = \vec{y}\,, \tag{6.1}$$

where:

- The λ_i functions are typically the so-called *Fully-Connected* (FC) layers and are expressible as affine functions: denoting $\vec{x} \in \mathbb{R}^D$ the input of an FC, its output is given by $\mathbf{A}\vec{x} + \vec{b}$, being $\mathbf{A} \in \mathbb{R}^{D \times C}$ a matrix of weights and $\vec{b} \in \mathbb{R}^C$ a vector of biases. These weights and biases are the trainable weights of the FC layer. They are called *Fully-Connected* because each i-th input coordinate is *connected* to each j-th output via the $\mathbf{A}[i, j]$ weight. FC layers can be seen as a special case of the linear layers in general feedforward networks, in which not all the connections are present. The absence of some (i, j)-th connections can be formalized as a constraint for the matrix $\mathbf{A}$ consisting in forcing to 0 its (i, j)-th coordinates.

- The σ_i are the so-called *activation functions* (ACT): an activation function is a non-linear real function that is applied independently to each coordinate of its input. In general it does not depend on trainable weights. We denote them by σ since in general they are functions similar to the *logistic sigmoid* introduced in 3.1.3 and denoted by σ as well: indeed historically sigmoidal functions where recommended, *i.e.* real-valued, bounded, monotonic, and differentiable functions with a non-negative first derivative. Nevertheless, the recommended function in modern neural network literature is the so-called *Rectified Linear Unit* (ReLU), introduced by [NH10] and defined as $\mathrm{ReLU}(\vec{x})[i] = \max(0, \vec{x}[i])$. Even it this function is not sigmoidal, not being bounded, nor differentiable, near transformation the fact of being a non-linear transformation but still piecewise linear, allows to preserve many of the properties that make linear models easy to optimize with gradient- function.

- s is the *softmax*[1] function (SOFT), already introduced in 3.1.3: $s(\vec{x})[i] = \frac{e^{\vec{x}[i]}}{\sum_j e^{\vec{x}[j]}}$.

The choice of the softmax function as last layer of a neural network classifier is the most common one. It allows the model $MLmodel$ to be interpreted as a generalisation of the binary classifier described in (3.13), where the softmax takes the place of the sigmoid to make the model multi-class and the linear argument is substituted by all previous layers of F. The previous layers takes in charge the feature extraction and preprocessing and are supposed to predict unnormalised log probabilities (3.9). The role of the *softmax* is thus to renormalise such output scores in such a way that they define a probability distribution $F(\vec{x}) \approx p_{Z \mid \vec{X}=\vec{x}}$.

6.3 Loss Function and Training

The weights of an NN are tuned during a training phase. They are first initialized with random values and are afterwards updated *via* an iterative approach which locally applies the (Stochastic) Gradient Descent algorithm [GBC16] to minimize a loss function quantifying the *classification error* of the function $F(\vec{X})$ over a training set which is a part of the profiling set.

6.3.1 Training

The training is said to be *full batch learning* if the full training database is processed before one update. At the opposite, if a single training input is processed at a time

[1] To prevent underflow, the log-softmax is usually preferred if several classification outputs must be combined.

then the approach is named *stochastic*. In practice, one often prefers to follow an approach in between, called *mini-batch learning*, and to use small *batch* (aka group) of training inputs at a time during the learning. In this case a step of the training consists in:

- selecting a batch of traces $(\vec{x}_i)_{i\in I}$ chosen in random order (here I is a random set of indexes),
- computing the outputs, or scores, of the current model function for the batch inputs $(\vec{y}_i = F(\vec{x}_i))_{i\in I}$,
- evaluating the loss function from the obtained scores and from the correct labels $(z_i)_{i\in I}$,
- compute the partial derivatives of the loss function with respect to each trainable weight (this is done through a method called *backpropagation* [**LeCun2012**]),
- updating trainable parameters by subtracting from each a small multiple of the loss gradient (the used multiple is called *learning rate*).

The size of the mini-batch is generally driven by several efficiency/accuracy factors which are *e.g.* discussed in [**GBC16**] (*e.g.* optimal use of the multi-core architectures, parallelisation with GPUs, trade-off between regularisation effect and stability, etc.).

An iteration over all the training datasets during the Stochastic Gradient Descent is called an *epoch*. The number of epochs is an important hyper-parameter to tune because small values may lead to underfitting and high values may lead to overfitting. In our experiments we chose to a so-called *early stopping* approach in order to avoid the need of a prior tuning of the number of epochs. It will be described in **??**.

6.3.2 Cross-Entropy

The cross-entropy [**LCH05**; GBC16] metric is a classical (and often by default) choice for the loss function in a classification-oriented NN. It is smooth and decomposable, and therefore amenable to optimization with standard gradient-based methods. However, other metrics may be investigated and can potentially lead to better results [**MHK10**; **SSZU15**].

Fro our experiments and simulations, we selected the the averaged *cross-entropy* between each $\vec{y}_i$ and the corresponding correct label $_i$ as loss function. Indeed nowadays this is the recommended choice for classification problems [GBC16]. There are two ways to interpret such a choice.

- First, recalling that $\vec{y}_i$ may be interpreted as an estimation of the conditional probability $\mathrm{Prob}[| =_i]$, the principle of maximum-likelihood suggests to drive the training in such a way that for such an estimate the probability of the correct label $_i$ is as high as possible. Thus, if we suppose the correct label vector $_i = (0, \dots, 0, \underbrace{1}_{j}, 0, \dots, 0)$ corresponds to the one-hot encoding of the label z^j, we want to maximize $\vec{y}_i[j]$ (or equivalently to minimize $-\log \vec{y}_i[j]$).[2] It may be

[2] We remark that thanks to the softmax function used as last network layer, each coordinate of $\vec{y}_i$ is always strictly positive.

observed that such a log-likelihood rewrites as

$$-\log \vec{y}_i[j] = -\sum_{t=1}^{|\mathcal{Z}|} {}_i[t] \log \vec{y}_i[t] \ . \tag{6.2}$$

- The second interpretation of the choice of the average cross-entropy comes from the observation that the vector ${}_i = (0, \ldots, 0, \underbrace{1}_{j}, 0, \ldots, 0)$ gives the value of the pmf of $| =_i$, which corresponds to the exact probability density we want the network to approximate. Informally speaking, the cross-entropy between two probability distributions ${}_i, \vec{y}_i$ gives a measure of dissimilarity between them, and is defined as follows:

$$\mathbb{H}({}_i, \vec{y}_i) = \mathbb{H}({}_i) + D_{KL}({}_i || \vec{y}_i) = \mathbb{E}_{{}_i}[-\log \vec{y}_i] = -\sum_{t=1}^{|\mathcal{Z}|} {}_i[t] \log \vec{y}_i[t] \ , \tag{6.3}$$

  where $\mathbb{H}$ denotes the entropy and D_{KL} denotes the Kullback-Leibler divergence (see [Bis06]-Sec. 1.6). Thus, this is an information-theoretic notion, that comes out to be equivalent to the negative log-likelihood formula given by (**??**).

In conclusion, depending on the point of view, minimizing the cross-entropy corresponds to maximize the likelihood of the right label, or to minimize the dissimilarity between the network estimation of a distribution and the right distribution that we want it to approximate. In practice, to train the neural network the loss function is given by the cross-entropy averaged over the traces contained in a batch:

$$\mathcal{L}(F, ({}_i)_{i \in I}) = \frac{1}{| I |} \sum_{i \in I} H({}_i, F({}_i)) \ . \tag{6.4}$$

A good choice for the size of the batch is a value as large as possible but which avoids computational performances drop. An iteration over the entire training set is called *epoch*. To monitor the training of an NN and to evaluate its performances it is a good practice to separate the labelled data into 3 sets:

- the proper *training set*, which is actually used to train the weights (in general it contains the greatest part of the labelled data)
- a *validation set*, which is observed in general at the end of each epoch to monitor the training
- a *test set*, which is kept unobserved during the training phase and which is involved to finally evaluate the performances of the trained NN.

For our experiments we will use the attack traces as test set, while we will split the profiling traces into a training set and a validation set.[3]

[3] The way how the profiling set is split into training and validation sets might induce a bias in the learned model. A good way to get rid of such a bias is to apply a *cross-validation* technique, *e.g.* a *10-fold cross-validation*. The latter one consists in partitioning the profiling set into 10 sub-sets, and in performing 10 times the training while choosing each time one of the sub-sets for the validation and the union of the 9 other ones for the training. An average over the performances of the 10 obtained models gives a more robust estimation of the accuracies and performances. Results of this papers do not make use of such a cross-validation technique.

6.3.2.1 The accuracy

is the most common metric to both monitor and evaluate an NN. It is defined as the successful classification rate reached over a dataset. The *training accuracy*, the *validation accuracy* and the *test accuracy* are the successful classification rates achieved respectively over the training, the validation and the test sets. At the end of each epoch it is useful to compute and to compare the training accuracy and the validation accuracy. For some trained models we will measure in this paper (see *e.g.* Table **??**) the following two additional quantities:

- the *maximal training accuracy*, corresponding to the maximum of the training accuracies computed at the end of each epoch
- the *maximal validation accuracy*, corresponding to the maximum of the validation accuracies computed at the end of each epoch.

In addition to the two quantities above, we will also evaluate the performances of our trained model, by computing a *test accuracy*. Sometimes it is useful to complete this evaluation by looking at the so-called *confusion matrix* (see the bottom part of Fig. **??**). Indeed the latter matrix enables, in case of misclassification, for the identification of the classes which are confused. The confusion matrix corresponds to the distribution over the couples *(true label, predicted label)* directly deduced from the results of the classification on the test set. A test accuracy of 100% corresponds to a diagonal confusion matrix.

6.3.2.2 On the Need to also Consider the Guessing Entropy.

The accuracy metric is perfectly adapted to the machine learning classification problem, but corresponds in side-channel language to the success rate of a Simple Attack, *i.e.* an attack where a single attack trace is available. When the attacker can acquire several traces for varying plaintexts, the accuracy metric is not sufficient alone to evaluate the attack performance. Indeed such a metric only takes into account the label corresponding to the maximal score and does not consider the other ones, whereas an SCA through (6.2) does (and therefore exploits the full information).

To take this remark into account, we will always associate the test accuracy to a side-channel metric defined as the minimal number $N^\star$ of side-channel traces that makes the *guessing entropy* (the average rank of the right key candidate) be permanently equal to 1 (see *e.g.* Table **??**). We will estimate such a guessing entropy through 10 independent attacks.

As we will see in the sections dedicated to our attack experiments, applying Machine Learning in a context where at the same time (1) the model to recover is complex and (2) the amount of exploitable measurements for the training is limited, may be ineffective due to some overfitting phenomena.

6.3.2.3 Overfitting.

Often the training accuracy is higher than the validation one. When the gap between the two accuracies is excessive, we assist to the *overfitting* phenomenon. It means that the NN is using its weights to *learn by heart* the training set instead of detecting significant discriminative features. For this reason its performances are poor over the validation set, which is new to it. Overfitting occurs when an NN is excessively

complex, *i.e.* when it is able to express an excessively large family of functions. In order to keep the NN as complex as wished and hence limiting the overfitting, some *regularization* techniques can be applied. For example, in this paper we will propose the use of the *Data Augmentation* (DA) [**simard2003best**] that consists in artificially adding observations to the training set. Moreover we will take advantage of the *early-stopping* technique [**Prechelt2012**] that consists in well choosing a stop condition based on the validation accuracy or on the validation loss (*i.e.* the value taken by the loss function over the validation set).

Several extensions and variants of the Stochastic Gradient Descent have been proposed in the context of DL. These variants, called *optimizers*, aim to adapt the *learning rate* (the step size) of the Gradient Descent during the training process. More details about the specification of neural networks will be given in the dedicated sections **??** and **??**, but we will not go further on the optimization approaches and the interested reader may refer to [GBC16].

6.4 Attack Strategy with an MLP

Looking back to the Template Attack strategy described in 2.2.3.1, we observe that the main difference between such a strategy and the exploitation of an MLP constructed as just described is the fact the the TA bases on a generative model, while MLPs are used to construct discriminative ones. Indeed, in TA the templates (2.9) are priorly estimated, while with an MLP $F(\vec{x}) \approx p_{Z \mid \vec{X}=\vec{x}}$ one approximates directly posterior probabilities (2.10). Once this approximation is done, the attack strategy proceeds in the same way for both approaches. The attacker acquires the new attack traces, that he only can associate to the public parameter E, obtaining couples $(\vec{x}_i, e_i)_{i=1,\dots,N_a}$. Then he makes key hypothesis $k \in \mathcal{K}$ and, making the assumption that each acquisition is an independent observation of $\vec{X}$, he associates to each hypothesis $k \in \mathcal{K}$ a score d_k given by (2.11), that in terms of MLP model F rewrites as:

$$d_k = \prod_{i=1}^{N_a} F(\vec{x}_i)[f(k, e_i)] \,. \tag{6.5}$$

Finally, the best key candidate $\hat{k}$ is the one maximizing such a joint probability, as in (2.12)

$$\hat{k} = \underset{k}{\operatorname{argmax}}\, d_k \,. \tag{6.6}$$

6.5 Shift-Invariance and Convolutional Neural Network

6.6 Data Augmentation for Misaligned Side-Channel Traces

6.7 Experiments against Software Countermeasures

6.8 Experiments against Artificial Hardware Countermeasures

6.9 Experiments against Real-Case Hardware Countermeasures

Chapter 7

Conclusions and Perspectives

7.1 Summary

7.2 Strengthen Embedded Security: the Main Challenge for Machine Learning Applications

Appendix A

Cross-Validation

In the Machine Learning community, several evaluation frameworks are commonly applied to assess the performances of a model or to select the best hyper-parameters for a learning algorithm. These methods aim to provide an estimator of the performance which does not depend on the choice of the training set $\mathcal{D}_{\text{train}}$ (on which the model is trained) and of the test set $\mathcal{D}_{\text{test}}$ (on which the model is tested) but only on their size.

The so-called *t-fold cross-validation* [FHT01] is currently the preferred evaluation method. Let P be a performance metric, $\hat{f}$ a model to evaluate, and $\mathcal{D}_{\text{train}} = (\vec{\mathcal{X}}, \mathcal{Y})$ a labelled dataset, the outline of the method is the following:

1. [optional] randomize the order of the labelled traces in $\mathcal{D}_{\text{train}}$,
2. split the samples and their corresponding labels into t disjoint parts of equal size $(\vec{\mathcal{X}}_1, \mathcal{Y}_1), \ldots, (\vec{\mathcal{X}}_t, \mathcal{Y}_t)$. For each $i \in [1..t]$, do:
   (a) set $\mathcal{D}_{\text{validation}} \doteq (\vec{\mathcal{X}}_i, \mathcal{Y}_i)$ and $\mathcal{D}_{\text{train}} \doteq (\bigcup_{j \neq i} \vec{\mathcal{X}}_j, \bigcup_{j \neq i} \mathcal{Y}_j)$,
   (b) (re-)train[1] the model $\hat{f}$ on $\mathcal{D}_{\text{train}}$,
   (c) compute the performance metric P_i by evaluating the model $\hat{f}$ on $\mathcal{D}_{\text{validation}}$,
3. return the mean $\frac{1}{t} \sum_{i=1}^{t} \mathrm{P}_i$.

It is known that the t-fold cross-validation estimator is an unbiased estimator of the generalization performance. Its main drawback is its variance which may be large and difficult to estimate [Bre+96; BG05].

[1] The model is trained from scratch at each iteration of the loop over t.

Appendix B

Artificially Simulate Jitter

In order to analyse the behaviour of the techniques studied in this thesis over misaligned side-channel traces, we simulated sometimes a jitter effect to misalign some well-synchronized traces in a controlled way. When jittering is present, the clock stability is altered and clock cycles are sampled by a varying number of time samples. To simulate such effect, the windows containing clock patterns of an acquisition are selected one by one and passed as input to the following function, described in python code, in charge to enlarge or reduce them in a random way. The randomness depends on two parameters `sigma` and `B`, being the number of inserted or removed points be almost normally distributed, with standard deviation given by `sigma`, but bounded. The bound is controlled by `B` by the following rule: the final size of a window has to be at least $\frac{1}{B}$ times the original size and at most `B` times the original size. The value assigned to newly inserted points is the linear interpolation of the previous and the following points.

```
def enlarge_reduce_window(window,sigma,B):
    Npts = window.shape[0]
    new_window = np.copy(window)
    deltaPts = int(np.floor(np.random.randn(1)[0]*sigma))
    if (deltaPts >= 0):
        deltaPts = min(Npts*(B-1),deltaPts)
        for i  in range(deltaPts):
            curr_size = new_window.shape[0]
            pos = int(np.floor(np.random.rand(1)*curr_size))
            if pos==0 or pos==curr_size-1:
                new_window = np.insert(new_window,
                  pos,new_window[pos])
            else:
                new_window = np.insert(new_window,pos,
                (new_window[pos-1]+
                new_window[pos])/2.0)
    else:
        deltaPts = max(-Npts*(1-1/B),deltaPts)
        for i in range(-deltaPts):
            curr_size = new_window.shape[0]
            pos = int(np.floor(np.random.rand(1)*curr_size))
            new_window = np.delete(new_window,pos)
    return new_window
```

This deformation is applied to each clock pattern independently. We remark that is implies that, for example, The 19th clock cycle of a deformed acquisition suffers from the cumulation of the previous 18 deformations. For the sake of visualizing the

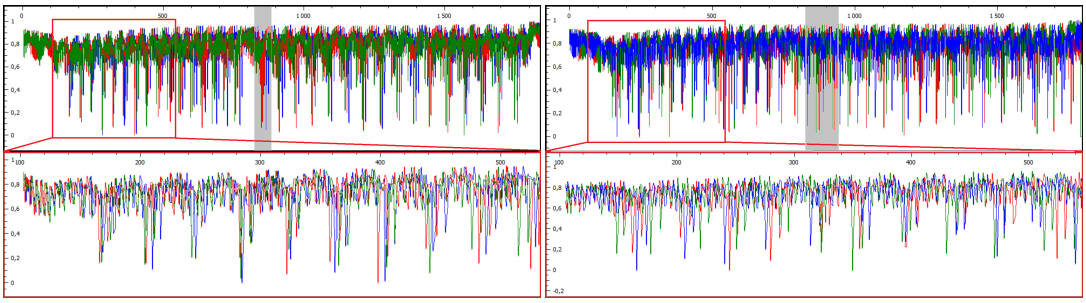


FIGURE B.1: Left: some traces of the *DS_low_jitter* dataset, a zoom of the part highlighted by the red rectangle is given in the bottom part. Right: some traces (and the relative) of the *DS_high_jitter* dataset. The interesting clock cycle for experiments in Sec. 6.8 is highlighted by the grey rectangular area.

effect of such a jitter simulation, in Fig. B.1 we depict some traces of *DS_low_jitter* and of the *DS_high_jitter* datasets, used for experiments in Sec .6.8. They are obtained by perfectly synchronous acquisitions, with parameters set to `sigma` $= 2$, `B`$= 2$ for the *DS_low_jitter* dataset and `sigma` $= 6$, `B`$= 6$ for the *DS_high_jitter* one.

Appendix C

Kernel PCA construction

Suppose that we want to perform PCA in the image space of a function Φ that is associated to a given kernel function K. The kernel version for PCA has been presented in [SSM98]; as we said in Chapter **??**, the important step consists in expressing the operations needed for the PCA procedure in terms of the dot products between the mapped data.

Let us assume that data are centered in the feature space, *i.e.* $\sum_{i=1,\dots,N_p} \Phi(\vec{x}_i) = 0$.[1] In this way the empirical covariance matrix $\mathbf{S}^\Phi$ of data in the feature space is given by:

$$\mathbf{S}^\Phi = \frac{1}{N_p} \sum_{i=1}^{N_p} \Phi(\vec{x}_i)\Phi(\vec{x}_i)^\intercal \,. \tag{C.1}$$

We want to find eigenvalues $\lambda^\Phi \neq 0$ and eigenvectors $\vec{\alpha}^\Phi \in \mathcal{F} \setminus \{\mathbf{0}\}$ such that

$$\mathbf{S}^\Phi \vec{\alpha}^\Phi = \lambda^\Phi \vec{\alpha}^\Phi \,. \tag{C.2}$$

We remark that such an eigenvector satisfies

$$\vec{\alpha}^\Phi = \frac{1}{\lambda^\Phi N_p} \sum_{i=1}^{N_p} \Phi(\vec{x}_i)\Phi(\vec{x}_i)^\intercal \vec{\alpha}^\Phi \tag{C.3}$$

$$= \frac{1}{\lambda^\Phi N_p} \sum_{i=1}^{N_p} \left[\Phi(\vec{x}_i)^\intercal \vec{\alpha}^\Phi\right] \Phi(\vec{x}_i) = \tag{C.4}$$

$$= \sum_{i=1}^{N_p} \underbrace{\frac{\Phi(\vec{x}_i)^\intercal \vec{\alpha}^\Phi}{\lambda^\Phi N_p}}_{\nu_i} \Phi(\vec{x}_i) = \tag{C.5}$$

$$= \sum_{i=1}^{N_p} \nu_i \Phi(\vec{x}_i) \,, \tag{C.6}$$

where the step (C.4) makes use of the associativity of the matrix product and the commutativity of the scalar-matrix product. Eq. (C.6) tells us that each eigenvector $\vec{\alpha}^\Phi$ is expressible as a linear combination of the data mapped into the feature space $(\Phi(\vec{x}_i)_{i=1,\dots,N_p}$, or equivalently each eigenvector $\vec{\alpha}^\Phi$ lies in the span of $(\Phi(\vec{x}_i)_{i=1,\dots,N_p})$.

[1] Such a condition is not hard to achieve, even without explicitly pass through the feature space: it suffices substituting the kernel matrix $\mathbf{K}$ by the matrix $\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_{N_p}\mathbf{K} - \mathbf{K}\mathbf{1}_{N_p} + \mathbf{1}_{N_p}\mathbf{K}\mathbf{1}_{N_p}$, where $\mathbf{1}_{N_p}$ denotes the matrix with each entry equal to $\frac{1}{N_p}$. The same kind of matrix has to be computed in projecting phase, using the test data.

This observation authorizes to substitute to the problem (C.2), the following equivalent system:

$$\begin{cases} \lambda^{\Phi}(\Phi(\vec{x}_1)\cdot\vec{\alpha}^{\Phi}) = \Phi(\vec{x}_1)\cdot\mathbf{S}^{\Phi}\vec{\alpha}^{\Phi} \\ \vdots \\ \lambda^{\Phi}(\Phi(\vec{x}_{N_p})\cdot\vec{\alpha}^{\Phi}) = \Phi(\vec{x}_{N_p})\cdot\mathbf{S}^{\Phi}\vec{\alpha}^{\Phi} \end{cases} \tag{C.7}$$

Joining (C.6) and (C.7) we obtain, looking to the first equation of the system:

$$\lambda^{\Phi}(\Phi(\vec{x}_1)\cdot\sum_{i=1}^{N_p}\nu_i\Phi(\vec{x}_i)) = \Phi(\vec{x}_1)\cdot\left[\frac{1}{N}\sum_{i=1}^{N_p}\Phi(\vec{x}_i)\Phi(\vec{x}_i)^{\intercal}(\sum_{i=1}^{N_p}\nu_i\Phi(\vec{x}_i))\right] \tag{C.8}$$

$$\lambda^{\Phi}\sum_{i=1}^{N_p}\nu_i(\Phi(\vec{x}_1)\cdot\Phi(\vec{x}_i)) = \Phi(\vec{x}_1)\cdot\left[\sum_{j=1}^{N_p}\frac{\nu_j}{N}\left(\sum_{i=1}^{N_p}\Phi(\vec{x}_i)\Phi(\vec{x}_i)^{\intercal}\right)\Phi(\vec{x}_j)\right] \tag{C.9}$$

$$\lambda^{\Phi}\sum_{i=1}^{N_p}\nu_i(\Phi(\vec{x}_1)\cdot\Phi(\vec{x}_i)) = \Phi(\vec{x}_1)\cdot\left[\sum_{j=1}^{N_p}\frac{\nu_j}{N}\sum_{i=1}^{N_p}\underbrace{\Phi(\vec{x}_i)^{\intercal}\Phi(\vec{x}_j)}_{\Phi(\vec{x}_i)\cdot\Phi(\vec{x}_j)}\Phi(\vec{x}_i)\right] \tag{C.10}$$

$$\lambda^{\Phi}\sum_{i=1}^{N_p}\nu_i(\Phi(\vec{x}_1)\cdot\Phi(\vec{x}_i)) = \sum_{j=1}^{N_p}\frac{\nu_j}{N}\left[\Phi(\vec{x}_1)\cdot\sum_{i=1}^{N_p}(\Phi(\vec{x}_i)\cdot\Phi(\vec{x}_j))\Phi(\vec{x}_i)\right] \tag{C.11}$$

$$N_p\lambda^{\Phi}\sum_{i=1}^{N_p}\nu_i\underbrace{(\Phi(\vec{x}_1)\cdot\Phi(\vec{x}_i))}_{\mathbf{K}[1,i]} = \sum_{j=1}^{N_p}\nu_j\left[\sum_{i=1}^{N_p}\underbrace{(\Phi(\vec{x}_i)\cdot\Phi(\vec{x}_j))}_{\mathbf{K}[i,j]}\underbrace{(\Phi(\vec{x}_1)\cdot\Phi(\vec{x}_i))}_{\mathbf{K}[1,j]}\right]. \tag{C.12}$$

Thus, the system (C.7) is equivalent to the follow:

$$\begin{cases} N_p\lambda^{\Phi}\sum_{i=1}^{N_p}\nu_i\mathbf{K}[1,i] & = \sum_{j=1}^{N_p}\nu_j\left[\sum_{i=1}^{N_p}\mathbf{K}[1,j]\mathbf{K}[i,j]\right] \\ \vdots & \\ N_p\lambda^{\Phi}\sum_{i=1}^{N_p}\nu_i\mathbf{K}[N_p,i] & = \sum_{j=1}^{N_p}\nu_j\left[\sum_{i=1}^{N_p}\mathbf{K}[N_p,j]\mathbf{K}[i,j]\right] \end{cases} \tag{C.13}$$

Let $\vec{\nu}$ be the column vector containing the coefficients ν_i of (C.6). The above system is expressible in matricial form as

$$\begin{cases} N_p\lambda^{\Phi}[\mathbf{K}\vec{\nu}][1] & = [\mathbf{K}^2\vec{\nu}][1] \\ \vdots & \\ N_p\lambda^{\Phi}[\mathbf{K}\vec{\nu}][N_p] & = [\mathbf{K}^2\vec{\nu}][N_p]\,, \end{cases} \tag{C.14}$$

which equals the following equation:

$$N_p\lambda^{\Phi}\mathbf{K}\vec{\nu} = \mathbf{K}^2\vec{\nu}\,. \tag{C.15}$$

It can be shown that solving the last equation is equivalent to solve the following eigenvector problem

$$\gamma\vec{\nu} = \mathbf{K}\vec{\nu}\,. \tag{C.16}$$

Let $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_{N_p}$ denote the eigenvalues of $\mathbf{K}$, γ_C being the last different from zero, and $\vec{\nu}_1,\ldots,\vec{\nu}_{N_p}$ the corresponding eigenvectors. For the sake of obtaining the corresponding normalized principal components in the feature space $\mathcal{F}$, denoted $\vec{\alpha}_1^{\Phi},\ldots,\vec{\alpha}_C^{\Phi}$, a normalization step is required, imposing for all $k = 1,\ldots,C$

$$\vec{\alpha}_k^{\Phi} \cdot \vec{\alpha}_k^{\Phi} = 1 \,, \tag{C.17}$$

which can be translated into a condition for $\vec{\nu}_1, \ldots, \vec{\nu}_C$, using (C.6) and (C.16):

$$1 = \sum_{i,j=1}^{N_p} \vec{\nu}_k[i]\vec{\nu}_k[j](\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)) = \vec{\nu}_k \cdot \mathbf{K}\vec{\nu}_k = \gamma_k(\vec{\nu}_k \cdot \vec{\nu}_k) \tag{C.18}$$

Extracting the non-linear principal components of a datum $\vec{x}$ means projecting its image $\Phi(\vec{x})$ onto the eigenvectors $\vec{\alpha}_1^{\Phi}, \ldots, \vec{\alpha}_C^{\Phi}$ in $\mathcal{F}$. To do so, we neither need to explicitly compute $\Phi(\vec{x})$ nor $\vec{\alpha}_i^{\Phi}$. Indeed, using (C.6):

$$\vec{\alpha}_k^{\Phi} \cdot \Phi(\vec{x}) = \sum_{i=1}^{N_p} \vec{\nu}_k[i](\Phi(\vec{x}_i) \cdot \Phi(\vec{x})) = \sum_{i=1}^{N_p} \vec{\nu}_k[i]K(\vec{x}_i, \vec{x}) \,. \tag{C.19}$$

C.1 Kernel class-oriented PCA

Suppose now that we want to perform a class-oriented PCA in the image space of a function Φ that is associated to a given kernel function K, i.e. we want to solve, using a kernel trick, the eigenvalue problem

$$\mathbf{S}_{\mathbf{B}}^{\Phi}\vec{\alpha}^{\Phi} = \lambda^{\Phi}\vec{\alpha}^{\Phi} \,, \tag{C.20}$$

where $\mathbf{S}_{\mathbf{B}}^{\Phi}$ is the between-scatter matrix in the feature space:

$$\mathbf{S}_{\mathbf{B}}^{\Phi} = \sum_{z \in \mathcal{Z}} N_z(\overline{\Phi(\vec{x})}^z - \overline{\Phi(\vec{x})})(\overline{\Phi(\vec{x})}^z - \overline{\Phi(\vec{x})})^{\intercal} \,. \tag{C.21}$$

Here $\overline{\Phi(\vec{x})}^z = \frac{1}{N_z}\sum_{i=1:\ z_i=z} \Phi(\vec{x}_i)$ and $\overline{\Phi(\vec{x})} = \frac{1}{N_p}\sum_{i=1}^{N_p} \Phi(\vec{x}_i)$.

As before, the eigenvectors $\vec{\alpha}_i^{\Phi}$ are expressible as linear combination of the data images on $\mathcal{F}$, i.e. (C.6) is still true:

$$\vec{\alpha}^{\Phi} = \sum_{i=1}^{N_p} \nu_i \Phi(\vec{x}_i) \,. \tag{C.22}$$

Moreover as before, the eigenvector problem (C.20) can be translated in an eigenvector problem that gives the coefficients $\vec{\nu}$ as solutions. That is:

$$\gamma\mathbf{M} = \mathbf{M}\vec{\nu} \,, \tag{C.23}$$

where the matrix $\mathbf{M}$ is computed as

$$\mathbf{M} = \sum_{z \in \mathcal{Z}} N_z(\vec{M}_z - \vec{M}_T)(\vec{M}_z - \vec{M}_T) \,, \tag{C.24}$$

where $\vec{M}_j = \frac{1}{N_z}\sum_{i=1:\ z_i=z} K(\vec{x}_j, \vec{x}i)$ is an N_p-sized column vector, and $\vec{M}_T[j] = \frac{1}{N_p}\sum_{i=1}^{N_p} K(\vec{x}_j, \vec{x}_i)$.

Finally, one the eigenvector $\vec{\nu}$ are found, to project a datum $\vec{x}$ onto the corresponding principal component in the feature space we proceed as in the previous case:

$$\vec{\alpha}_k^{\Phi} \cdot \Phi(\vec{x}) = \sum_{i=1}^{N_p} \vec{\nu}_k[i] K(\vec{x}_i, \vec{x}) \; . \tag{C.25}$$

Bibliography

[Abi] *8.8 Billion Smart Cards Shipped in 2014 Driven by Growth in the Banking and SIM Card Markets*. https://www.abiresearch.com/press/88-billion-smart-cards-shipped-in-2014-driven-by-g/. Accessed: 2018-07-02 (cit. on p. 7).

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. "An implementation of DES and AES, secure against some attacks". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2001, pp. 309–318 (cit. on p. 31).

[Arc+06] C. Archambeau et al. "Template Attacks in Principal Subspaces". English. In: *Cryptographic Hardware and Embedded Systems - CHES 2006*. Ed. by Louis Goubin and Mitsuru Matsui. Vol. 4249. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 1–14. ISBN: 978-3-540-46559-1. DOI: 10.1007/11894063_1. URL: http://dx.doi.org/10.1007/11894063_1 (cit. on pp. 45, 46, 50).

[BA00] Gaston Baudat and Fatiha Anouar. "Generalized discriminant analysis using a kernel approach". In: *Neural computation* 12.10 (2000), pp. 2385–2404 (cit. on p. 70).

[Bar+15] Gilles Barthe et al. "Verified Proofs of Higher-Order Masking". In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*. 2015, pp. 457–485. DOI: 10.1007/978-3-662-46800-5_18. URL: https://doi.org/10.1007/978-3-662-46800-5_18 (cit. on p. 16).

[Bat+11] Lejla Batina et al. "Mutual information analysis: a comprehensive study". In: *Journal of Cryptology* 24.2 (2011), pp. 269–291 (cit. on pp. 24, 32).

[Bat+16] Alberto Battistello et al. "Horizontal side-channel attacks and countermeasures on the ISW masking scheme". In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2016, pp. 23–39 (cit. on p. 23).

[Bau+13] Aurélie Bauer et al. "Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations." In: *CT-RSA*. Springer. 2013, pp. 1–17 (cit. on p. 23).

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. "Correlation power analysis with a leakage model". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2004, pp. 16–29 (cit. on p. 24).

[BDP10] Martin Bär, Hermann Drexler, and Jürgen Pulkus. "Improved template attacks". In: *COSADE2010* (2010) (cit. on p. 30).

[BG05] Yoshua Bengio and Yves Grandvalet. "Bias in estimating the variance of K-fold cross-validation". In: *Statistical modeling and analysis for complex data problems*. Springer, 2005, pp. 75–95 (cit. on p. 87).

[BGK04] Johannes Blömer, Jorge Guajardo, and Volker Krummel. "Provably secure masking of AES". In: *International Workshop on Selected Areas in Cryptography*. Springer. 2004, pp. 69–83 (cit. on p. 31).

[BHK97] Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. 1997 (cit. on pp. 46, 56).

[BHW12] 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". English. In: *Topics in Cryptology CT-RSA 2012*. Ed. by Orr Dunkelman. Vol. 7178. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 383–397. ISBN: 978-3-642-27953-9. DOI: 10.1007/978-3-642-27954-6_24. URL: http://dx.doi.org/10.1007/978-3-642-27954-6_24 (cit. on pp. 45, 46, 51).

[Bil+14] Begül Bilgin et al. "A more efficient AES threshold implementation". In: *International Conference on Cryptology in Africa*. Springer. 2014, pp. 267–284 (cit. on p. 31).

[Bis06] Christopher M.. Bishop. *Pattern recognition and machine learning*. Springer, 2006 (cit. on pp. 27, 28, 35, 47, 78).

[BLR13] Timo Bartkewitz and Kerstin Lemke-Rust. "Efficient Template Attacks Based on Probabilistic Multi-class Support Vector Machines". English. In: *Smart Card Research and Advanced Applications*. Ed. by Stefan Mangard. Vol. 7771. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 263–276. ISBN: 978-3-642-37287-2. DOI: 10.1007/978-3-642-37288-9_18. URL: http://dx.doi.org/10.1007/978-3-642-37288-9_18 (cit. on p. 41).

[Bre+96] Leo Breiman et al. "Heuristics of instability and stabilization in model selection". In: *The annals of statistics* 24.6 (1996), pp. 2350–2383 (cit. on p. 87).

[Bru+ a] N. Bruneau et al. "Less is more: Dimensionality reduction from a theoretical perspective". In: *17th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2015)*. to appear, 2015 (cit. on pp. 45, 54).

[Bru+14] Nicolas Bruneau et al. "Boosting Higher-Order Correlation Attacks by Dimensionality Reduction". English. In: *Security, Privacy, and Applied Cryptography Engineering*. Ed. by RajatSubhra Chakraborty, Vashek Matyas, and Patrick Schaumont. Vol. 8804. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 183–200. ISBN: 978-3-319-12059-1. DOI: 10.1007/978-3-319-12060-7_13. URL: http://dx.doi.org/10.1007/978-3-319-12060-7_13 (cit. on p. 68).

[Car+14] Claude Carlet et al. "Achieving side-channel high-order correlation immunity with leakage squeezing". In: *Journal of Cryptographic Engineering* 4.2 (2014), pp. 107–121 (cit. on pp. 32, 66).

[CDP16a] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. "Enhancing Dimensionality Reduction Methods for Side-Channel Attacks". In: *Smart Card Research and Advanced Applications: 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*. Ed. by Naofumi Homma and Marcel Medwed. Cham: Springer International Publishing, 2016, pp. 15–33. ISBN: 978-3-319-31271-2. DOI: 10.1007/978-3-319-31271-2_2. URL: http://dx.doi.org/10.1007/978-3-319-31271-2_2 (cit. on p. 45).

[CDP16b] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. "Kernel Discriminant Analysis for Information Extraction in the Presence of Masking". In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2016, pp. 1–22 (cit. on pp. 17, 65).

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. "Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing". In: *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. Lecture Notes in Computer Science. Springer, 2017, pp. 45–68. ISBN: 978-3-319-66786-7. DOI: 10.1007/978-3-319-66787-4_3. URL: https://doi.org/10.1007/978-3-319-66787-4_3 (cit. on pp. 18, 81).

[CG15] John P Cunningham and Zoubin Ghahramani. "Linear dimensionality reduction: survey, insights, and generalizations." In: *Journal of Machine Learning Research* 16.1 (2015), pp. 2859–2900 (cit. on p. 45).

[Cha+99] Suresh Chari et al. "Towards sound approaches to counteract power-analysis attacks". In: *Annual International Cryptology Conference*. Springer. 1999, pp. 398–412 (cit. on pp. 26, 31, 66, 67).

[Che+00] Li-Fen Chen et al. "A new LDA-based face recognition system which can solve the small sample size problem". In: *Pattern Recognition* 33.10 (2000), pp. 1713 –1726. ISSN: 0031-3203. DOI: http://dx.doi.org/10.1016/S0031-3203(99)00139-9. URL: http://www.sciencedirect.com/science/article/pii/S0031320399001399 (cit. on pp. 46, 57).

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. "An efficient method for random delay generation in embedded software". In: *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 156–170 (cit. on p. 31).

[CK10] Jean-Sébastien Coron and Ilya Kizhvatov. "Analysis and improvement of the random delay countermeasure of CHES 2009". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2010, pp. 95–109 (cit. on p. 31).

[CK14a] Omar Choudary and Markus G Kuhn. "Efficient Stochastic Methods: Profiled Attacks Beyond 8 Bits". In: *IACR Cryptology ePrint Archive* (2014). URL: http://eprint.iacr.org/2014/885.pdf (cit. on p. 45).

[CK14b] Omar Choudary and Markus G Kuhn. "Efficient template attacks". In: *Smart Card Research and Advanced Applications*. Springer, 2014, pp. 253–270 (cit. on pp. 28, 45, 46, 51).

[CK14c] Omar Choudary and Markus G Kuhn. "Template attacks on different devices". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2014, pp. 179–198 (cit. on p. 62).

[CK18] Marios O Choudary and Markus G Kuhn. "Efficient, Portable Template Attacks". In: *IEEE Transactions on Information Forensics and Security* 13.2 (2018), pp. 490–501 (cit. on p. 46).

[CL06] Tonatiuh Peña Centeno and Neil D Lawrence. "Optimising kernel parameters and regularisation coefficients for non-linear discriminant analysis". In: *The Journal of Machine Learning Research* 7 (2006), pp. 455–491 (cit. on p. 74).

[CMW14] Christophe Clavier, Damien Marion, and Antoine Wurcker. "Simple power analysis on AES key expansion revisited". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2014, pp. 279–297 (cit. on p. 23).

[Cor+13] Jean-Sébastien Coron et al. "Higher-order side channel security and mask refreshing". In: *International Workshop on Fast Software Encryption*. Springer. 2013, pp. 410–424 (cit. on p. 31).

[CRR03] Suresh Chari, JosyulaR. Rao, and Pankaj Rohatgi. "Template Attacks". English. In: *Cryptographic Hardware and Embedded Systems - CHES 2002*. Ed. by Burton S. Kaliski, Cetin K. Koc, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 13–28. ISBN: 978-3-540-00409-7. DOI: 10.1007/3-540-36400-5_3. URL: http://dx.doi.org/10.1007/3-540-36400-5_3 (cit. on pp. 27, 28, 29).

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297 (cit. on p. 41).

[DC+15] Thomas De Cnudde et al. "Higher-order threshold implementation of the AES S-box". In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2015, pp. 259–272 (cit. on p. 31).

[DS15] François Durvaux and François-Xavier Standaert. "From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces". In: *IACR Cryptology ePrint Archive* (2015), p. 536 (cit. on p. 68).

[Dur+15] François Durvaux et al. "Efficient selection of time samples for higher-order DPA with projection pursuits". In: *Constructive Side-Channel Analysis and Secure Design*. Springer, 2015, pp. 34–50 (cit. on pp. 46, 67).

[EPW10] Thomas Eisenbarth, Christof Paar, and Bjorn Weghenkel. "Building a Side Channel Based Disassembler". English. In: *Transactions on Computational Science X*. Ed. by MarinaL. Gavrilova, C.J.Kenneth Tan, and EdwardDavid Moreno. Vol. 6340. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 78–99. ISBN: 978-3-642-17498-8. DOI: 10.1007/978-3-642-17499-5_4. URL: http://dx.doi.org/10.1007/978-3-642-17499-5_4 (cit. on p. 45).

[Fel08] Willliam Feller. *An introduction to probability theory and its applications*. Vol. 2. John Wiley & Sons, 2008 (cit. on p. 20).

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, 2001 (cit. on p. 87).

[FT74] Jerome H Friedman and John W Tukey. "A projection pursuit algorithm for exploratory data analysis". In: *IEEE Transactions on computers* 100.9 (1974), pp. 881–890 (cit. on p. 46).

[Fuk90] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition (2Nd Ed.)* San Diego, CA, USA: Academic Press Professional, Inc., 1990. ISBN: 0-12-269851-7 (cit. on pp. 54, 56).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on p. 82).

[Gen+15] Daniel Genkin et al. "Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2015, pp. 207–228 (cit. on p. 8).

[Gen+16] Daniel Genkin et al. "ECDH key-extraction via low-bandwidth electromagnetic attacks on PCs". In: *Cryptographers' Track at the RSA Conference*. Springer. 2016, pp. 219–235 (cit. on p. 8).

[Gie+10] Benedikt Gierlichs et al. "Revisiting higher-order DPA attacks". In: *Cryptographers? Track at the RSA Conference*. Springer. 2010, pp. 221–234 (cit. on p. 32).

[GLRP06] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. "Templates vs. stochastic methods". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2006, pp. 15–29 (cit. on pp. 29, 30).

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. "Electromagnetic analysis: Concrete results". In: *Cryptographic Hardware and Embedded Systems - CHES 2001*. Springer. 2001, pp. 251–261 (cit. on p. 8).

[GP99] Louis Goubin and Jacques Patarin. "DES and differential power analysis the ?Duplication? method". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 1999, pp. 158–172 (cit. on p. 31).

[GPT15] Daniel Genkin, Itamar Pipman, and Eran Tromer. "Get your hands off my laptop: Physical side-channel key-extraction attacks on PCs". In: *Journal of Cryptographic Engineering* 5.2 (2015), pp. 95–112 (cit. on p. 8).

[GR17] Dahmun Goudarzi and Matthieu Rivain. "How Fast Can Higher-Order Masking Be in Software?" In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2017, pp. 567–597 (cit. on p. 31).

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. "RSA key extraction via low-bandwidth acoustic cryptanalysis". In: *International Cryptology Conference*. Springer. 2014, pp. 444–461 (cit. on p. 8).

[Hos+11] Gabriel Hospodar et al. "Machine learning in side-channel analysis: a first study". English. In: *Journal of Cryptographic Engineering* 1.4 (2011), pp. 293–302. ISSN: 2190-8508. DOI: 10.1007/s13389-011-0023-x. URL: http://dx.doi.org/10.1007/s13389-011-0023-x (cit. on p. 41).

[Hua+02] Rui Huang et al. “Solving the small sample size problem of LDA”. In: *Pattern Recognition, 2002. Proceedings. 16th International Conference on*. Vol. 3. 2002, 29–32 vol.3. DOI: 10.1109/ICPR.2002.1047787 (cit. on pp. 46, 57).

[HZ12] Annelie Heuser and Michael Zohner. “Intelligent Machine Homicide”. English. In: *Constructive Side-Channel Analysis and Secure Design*. Ed. by Werner Schindler and SorinA. Huss. Vol. 7275. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 249–264. ISBN: 978-3-642-29911-7. DOI: 10.1007/978-3-642-29912-4_18. URL: http://dx.doi.org/10.1007/978-3-642-29912-4_18 (cit. on p. 41).

[IPS02] James Irwin, Dan Page, and Nigel P Smart. “Instruction stream mutation for non-deterministic processors”. In: *Application-Specific Systems, Architectures and Processors, 2002. Proceedings. The IEEE International Conference on*. IEEE. 2002, pp. 286–295 (cit. on p. 31).

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. “Private circuits: Securing hardware against probing attacks”. In: *Annual International Cryptology Conference*. Springer. 2003, pp. 463–481 (cit. on pp. 16, 25, 31).

[JPS05] Marc Joye, Pascal Paillier, and Berry Schoenmakers. “On second-order differential power analysis”. In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2005, pp. 293–308 (cit. on p. 32).

[JS17] Anthony Journault and François-Xavier Standaert. “Very high order masking: Efficient implementation and security evaluation”. In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2017, pp. 623–643 (cit. on p. 31).

[Kar+09] Peter Karsmakers et al. *Side channel attacks on cryptographic devices as a classification problem*. Tech. rep. COSIC technical report, 2009 (cit. on p. 45).

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. “Differential power analysis”. In: *Annual International Cryptology Conference*. Springer. 1999, pp. 388–397 (cit. on p. 8).

[Koc96] Paul C Kocher. “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”. In: *Annual International Cryptology Conference*. Springer. 1996, pp. 104–113 (cit. on pp. 7, 8).

[LBM14] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. “Power analysis attack: an approach based on machine learning”. In: *International Journal of Applied Cryptography* 3.2 (2014), pp. 97–115 (cit. on p. 41).

[LBM15] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. “A machine learning approach against a masked AES”. In: *Journal of Cryptographic Engineering* 5.2 (2015), pp. 123–139 (cit. on p. 41).

[LCY92] Ke Liu, Yong-Qing Cheng, and Jing-Yu Yang. “A generalized optimal set of discriminant vectors”. In: *Pattern Recognition* 25.7 (1992), pp. 731–739 (cit. on p. 57).

[Ler+15] Liran Lerman et al. “Template Attacks vs. Machine Learning Revisited (and the Curse of Dimensionality in Side-Channel Analysis)”. In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2015, pp. 20–33 (cit. on p. 41).

[Lib13] Joint Interpretation Library. *Application of Attack Potential to SmartSmart, Version 2.9*. Tech. rep. Common Criteria, 2013 (cit. on p. 14).

[Lio+14] Rokach Lior et al. *Data mining with decision trees: theory and applications*. Vol. 81. World scientific, 2014 (cit. on p. 41).

[Lom+14] Victor Lomné et al. "How to Estimate the Success Rate of Higher-Order Side-Channel Attacks". English. In: *Cryptographic Hardware and Embedded Systems CHES 2014*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 35–54. ISBN: 978-3-662-44708-6. DOI: 10.1007/978-3-662-44709-3_3. URL: http://dx.doi.org/10.1007/978-3-662-44709-3_3 (cit. on p. 67).

[LPR13] Victor Lomné, Emmanuel Prouff, and Thomas Roche. "Behind the scene of side channel attacks". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2013, pp. 506–525 (cit. on p. 29).

[LRP07] Kerstin Lemke-Rust and Christof Paar. *Gaussian mixture models for higher-order side channel analysis*. Springer, 2007 (cit. on p. 67).

[LZO06] Tao Li, Shenghuo Zhu, and Mitsunori Ogihara. "Using discriminant analysis for multi-class classification: an experimental investigation". In: *Knowledge and Information Systems* 10.4 (2006), pp. 453–472. ISSN: 0219-3116. DOI: 10.1007/s10115-006-0013-y. URL: http://dx.doi.org/10.1007/s10115-006-0013-y (cit. on p. 74).

[Man02] Stefan Mangard. "A simple power-analysis (SPA) attack on implementations of the AES key expansion". In: *International Conference on Information Security and Cryptology*. Springer. 2002, pp. 343–358 (cit. on p. 23).

[Man04] Stefan Mangard. "Hardware countermeasures against DPA–a statistical analysis of their effectiveness". In: *Topics in Cryptology–CT-RSA 2004*. Springer, 2004, pp. 222–235 (cit. on p. 31).

[Mas94] James L Massey. "Guessing and entropy". In: *Information Theory, 1994. Proceedings., 1994 IEEE International Symposium on*. IEEE. 1994, p. 204 (cit. on p. 26).

[Mav+12] Dimitrios Mavroeidis et al. "PCA, Eigenvector Localization and Clustering for Side-Channel Attacks on Cryptographic Hardware Devices". English. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by PeterA. Flach, Tijl De Bie, and Nello Cristianini. Vol. 7523. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 253–268. ISBN: 978-3-642-33459-7. DOI: 10.1007/978-3-642-33460-3_22. URL: http://dx.doi.org/10.1007/978-3-642-33460-3_22 (cit. on pp. 46, 51, 52).

[MDM16] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. "Profiling power analysis attack based on MLP in DPA contest V4. 2". In: *Telecommunications and Signal Processing (TSP), 2016 39th International Conference on*. IEEE. 2016, pp. 223–226 (cit. on p. 42).

[Mes00a] Thomas S Messerges. "Securing the AES finalists against power analysis attacks". In: *International Workshop on Fast Software Encryption*. Springer. 2000, pp. 150–164 (cit. on p. 31).

[Mes00b] Thomas S Messerges. "Using second-order power analysis to attack DPA resistant software". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2000, pp. 238–251 (cit. on p. 32).

[MHM13] Zdenek Martinasek, Jan Hajny, and Lukas Malina. "Optimization of power analysis using neural network". In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2013, pp. 94–107 (cit. on p. 42).

[MMS01] David May, Henk L Muller, and Nigel P Smart. "Non-deterministic processors". In: *Australasian Conference on Information Security and Privacy*. Springer. 2001, pp. 115–129 (cit. on p. 31).

[MMT15] Zdenek Martinasek, Lukas Malina, and Krisztina Trasy. "Profiling power analysis attack based on multi-layer perceptron network". In: *Computational Problems in Science and Engineering*. Springer, 2015, pp. 317–339 (cit. on p. 42).

[Moo+02] Simon Moore et al. "Improving smart card security using self-timed circuits". In: *Asynchronous Circuits and Systems, 2002. Proceedings. Eighth International Symposium on*. IEEE. 2002, pp. 211–218 (cit. on p. 31).

[Moo+03] Simon Moore et al. "Balanced self-checking asynchronous logic for smart card applications". In: *Microprocessors and Microsystems* 27.9 (2003), pp. 421–430 (cit. on p. 31).

[MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*. Vol. 31. Springer Science & Business Media, 2008 (cit. on pp. 25, 29, 31).

[Mor+11] Amir Moradi et al. "Pushing the limits: a very compact and a threshold implementation of AES". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2011, pp. 69–88 (cit. on p. 31).

[Mor74] Roland Moreno. *Methods of data storage and data storage systems*. US3971916A. 1974 (cit. on p. 7).

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. "Breaking Cryptographic Implementations Using Deep Learning Techniques". In: *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer. 2016, pp. 3–26 (cit. on pp. 42, 82).

[MZ13] Zdenek Martinasek and Vaclav Zeman. "Innovative method of the power analysis". In: *Radioengineering* 22.2 (2013), pp. 586–594 (cit. on p. 42).

[Nag+07] Sei Nagashima et al. "DPA using phase-based waveform matching against random-delay countermeasure". In: *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE. 2007, pp. 1807–1810 (cit. on p. 31).

[NH10] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814 (cit. on p. 83).

[NIS] FIPS PUB NIST. *197," Advanced Encryption Standard (AES)," November 2001* (cit. on pp. 4, 5, 6).

[OC14] Colin O'Flynn and Zhizhang David Chen. "ChipWhisperer: An open-source platform for hardware embedded security research". In: *Constructive Side-Channel Analysis and Secure Design*. Springer, 2014, pp. 243–260 (cit. on pp. 58, 72).

[Osw+05] Elisabeth Oswald et al. "A side-channel analysis resistant description of the AES S-box". In: *International Workshop on Fast Software Encryption*. Springer. 2005, pp. 413–423 (cit. on p. 31).

[Osw+06] Elisabeth Oswald et al. "Practical second-order DPA attacks for masked smart card implementations of block ciphers". In: *Cryptographers' Track at the RSA Conference*. Springer. 2006, pp. 192–207 (cit. on p. 32).

[Ou+17] Changhai Ou et al. *Manifold Learning Towards Masking Implementations: A First Study*. Cryptology ePrint Archive, Report 2017/1112. https://eprint.iacr.org/2017/1112. 2017 (cit. on p. 65).

[OWW14] Yossef Oren, Ofir Weisse, and Avishai Wool. "A New Framework for Constraint-Based Probabilistic Template Side Channel Attacks". English. In: *Cryptographic Hardware and Embedded Systems CHES 2014*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 17–34. ISBN: 978-3-662-44708-6. DOI: 10.1007/978-3-662-44709-3_2. URL: http://dx.doi.org/10.1007/978-3-662-44709-3_2 (cit. on p. 25).

[Par] TELECOM ParisTech. "DPA Contest 4". In: (). http://www.DPAcontest.org/v4/. URL: http://www.DPAcontest.org/v4/ (cit. on p. 51).

[Pee+05] Eric Peeters et al. "Improved higher-order side-channel attacks with FPGA experiments". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2005, pp. 309–323 (cit. on p. 32).

[PHG17] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. "Template attack versus Bayes classifier". In: *Journal of Cryptographic Engineering* 7.4 (2017), pp. 343–351 (cit. on p. 28).

[PM05] Thomas Popp and Stefan Mangard. "Masked dual-rail pre-charge logic: DPA-resistance without routing constraints". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2005, pp. 172–186 (cit. on p. 30).

[PR13] Emmanuel Prouff and Matthieu Rivain. "Masking against side-channel attacks: A formal security proof". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2013, pp. 142–159 (cit. on pp. 16, 26, 31).

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. "Statistical Analysis of Second Order Differential Power Analysis". In: *IEEE Trans. Computers* 58.6 (2009), pp. 799–811. DOI: 10.1109/TC.2009.15. URL: http://doi.ieeecomputersociety.org/10.1109/TC.2009.15 (cit. on pp. 32, 66).

[Pro+18] Emmanuel Prouff et al. *Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database*. Cryptology ePrint Archive, Report 2018/053. https://eprint.iacr.org/2018/053. 2018 (cit. on p. 82).

[PS08] Gilles Piret and F-X Standaert. "Security analysis of higher-order Boolean masking schemes for block ciphers (with conditions of perfect masking)". In: *IET Information Security* 2.1 (2008), pp. 1–11 (cit. on p. 32).

[QS01] Jean-Jacques Quisquater and David Samyde. "Electromagnetic analysis (ema): Measures and counter-measures for smart cards". In: *Smart Card Programming and Security* (2001), pp. 200–210 (cit. on p. 8).

[RGV12] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. "Selecting Time Samples for Multivariate DPA Attacks". English. In: *Cryptographic Hardware and Embedded Systems CHES 2012*. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 155–174. ISBN: 978-3-642-33026-1. DOI: 10.1007/978-3-642-33027-8_10. URL: http://dx.doi.org/10.1007/978-3-642-33027-8_10 (cit. on p. 67).

[Riv91] Ronald L Rivest. "Cryptography and machine learning". In: *International Conference on the Theory and Application of Cryptology*. Springer. 1991, pp. 427–439 (cit. on p. 41).

[RO05] Christian Rechberger and Elisabeth Oswald. "Practical Template Attacks". English. In: *Information Security Applications*. Ed. by ChaeHoon Lim and Moti Yung. Vol. 3325. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 440–456. ISBN: 978-3-540-24015-0. DOI: 10.1007/978-3-540-31815-6_35. URL: http://dx.doi.org/10.1007/978-3-540-31815-6_35 (cit. on p. 29).

[RP10] Matthieu Rivain and Emmanuel Prouff. "Provably secure higher-order masking of AES". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2010, pp. 413–427 (cit. on p. 31).

[RS09] Mathieu Renauld and François-Xavier Standaert. "Algebraic Side-Channel Attacks." In: *Inscrypt* 6151 (2009), pp. 393–410 (cit. on p. 24).

[RSVC09] Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. "Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA." In: *CHES*. Vol. 5747. Springer. 2009, pp. 97–111 (cit. on p. 24).

[SA08] François-Xavier Standaert and Cedric Archambeau. "Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages". English. In: *Cryptographic Hardware and Embedded Systems CHES 2008*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 411–425. ISBN: 978-3-540-85052-6. DOI: 10.1007/978-3-540-85053-3_26. URL: http://dx.doi.org/10.1007/978-3-540-85053-3_26 (cit. on pp. 45, 54, 56).

[SM99] Bernhard Schölkopf and Klaus-Robert Mullert. "Fisher discriminant analysis with kernels". In: *Neural networks for signal processing IX* 1 (1999), p. 1 (cit. on pp. 70, 73).

[SMY09] François-Xavier Standaert, TalG. Malkin, and Moti Yung. "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks". English. In: *Advances in Cryptology - EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 443–461. ISBN: 978-3-642-01000-2. DOI: 10.1007/978-3-642-01001-9_26. URL: http://dx.doi.org/10.1007/978-3-642-01001-9_26 (cit. on p. 26).

[SP06] Kai Schramm and Christof Paar. "Higher order masking of the AES". In: *Cryptographers? Track at the RSA Conference*. Springer. 2006, pp. 208–225 (cit. on p. 31).

[Spe+15] Robert Specht et al. "Improving Non-Profiled Attacks on Exponentiations Based on Clustering and Extracting Leakage from Multi-Channel High-Resolution EM Measurements". In: *Sixth International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2015)*. 2015 (cit. on pp. 46, 51).

[SPQ05] F-X Standaert, Eric Peeters, and J-J Quisquater. "On the masking countermeasure and higher-order power analysis attacks". In: *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*. Vol. 1. IEEE. 2005, pp. 562–567 (cit. on p. 32).

[SSM98] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear component analysis as a kernel eigenvalue problem". In: *Neural computation* 10.5 (1998), pp. 1299–1319 (cit. on pp. 70, 91).

[TM97] Mitchell T. M. *Machine Learning*. McGraw-Hill, New York, 1997 (cit. on p. 33).

[Ugo77] Michel Ugon. *Portable data carrier including a microprocessor*. US4211919A. 1977 (cit. on p. 7).

[VC+12] Nicolas Veyrat-Charvillon et al. "Shuffling against side-channel attacks: A comprehensive study with cautionary note". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2012, pp. 740–757 (cit. on p. 31).

[VGS14] Nicolas Veyrat-Charvillon, Benoit Gérard, and François-Xavier Standaert. "Soft Analytical Side-Channel Attacks". In: *IACR Cryptology ePrint Archive* 2014 (2014), p. 410. URL: https://eprint.iacr.org/2014/410.pdf (cit. on p. 24).

[WM97] David H Wolpert and William G Macready. "No free lunch theorems for optimization". In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82 (cit. on p. 41).

[WW04] Jason Waddle and David Wagner. "Towards Efficient Second-Order Power Analysis". English. In: *Cryptographic Hardware and Embedded Systems - CHES 2004*. Ed. by Marc Joye and Jean-Jacques Quisquater. Vol. 3156. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 1–15. ISBN: 978-3-540-22666-6. DOI: 10.1007/978-3-540-28632-5_1. URL: http://dx.doi.org/10.1007/978-3-540-28632-5_1 (cit. on p. 32).

[WW98] Jason Weston and Chris Watkins. *Multi-class support vector machines*. Tech. rep. Citeseer, 1998 (cit. on p. 41).

[WWB11] Jasper GJ van Woudenberg, Marc F Witteman, and Bram Bakker. "Improving differential power analysis by elastic alignment". In: *Cryptographersí Track at the RSA Conference*. Springer. 2011, pp. 104–119 (cit. on p. 31).

[YY01] Hua Yu and Jie Yang. "A direct LDA algorithm for high-dimensional data with application to face recognition". In: *Pattern Recognition* 34 (2001), pp. 2067–2070 (cit. on pp. 46, 57).

[Zho+17] Xinping Zhou et al. "A Novel Use of Kernel Discriminant Analysis as a Higher-Order Side-Channel Distinguisher". In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2017, pp. 70–87 (cit. on pp. 65, 78).