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

January 24, 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

List of Symbols

$GF(2^b)$ Galois Field of order 2^b

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*, denotes 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. 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 parallels, 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 be 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 functionality 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 private 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 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 standardized 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.

In the following part of this section a description of the two standard cryptographic primitives, *i.e.* building block algorithms used to build cryptographic protocols, that will be used in this thesis will; a symmetric one, the AES, and an asymmetric one, the RSA.

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 symmetric *block cipher*, *i.e.* an algorithm operating on fixed-length groups of bits.[1] The AES operates on blocks of 128 bits of plaintext, and can use keys of size 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 unit for processing in the AES algorithm is a byte. For AES internal operations, bytes are arranged on a two-dimensional array of bytes 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, ShiftRow and MixColumn. 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
ShiftRows
MixColumns
AddRoundKey

Last Round: SubBytes
ShiftRows
AddRoundKey

A description of the five functions is provided hereafter.

[1] in contrast with *stream ciphers*, which operate over a single plaintext bit at time

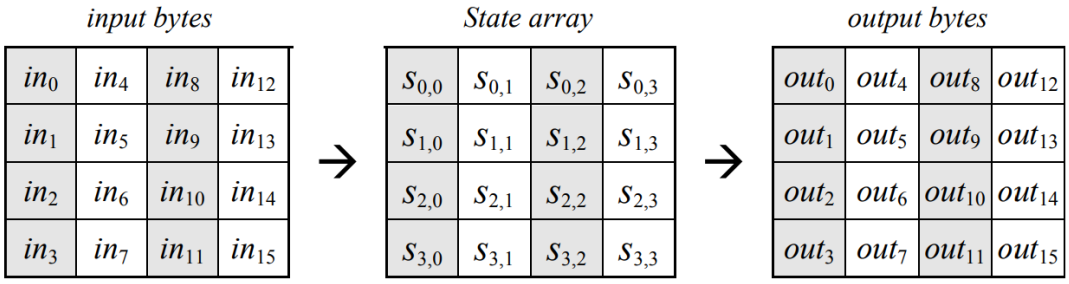


FIGURE 1.1: State array input and output. Source: [NIS].

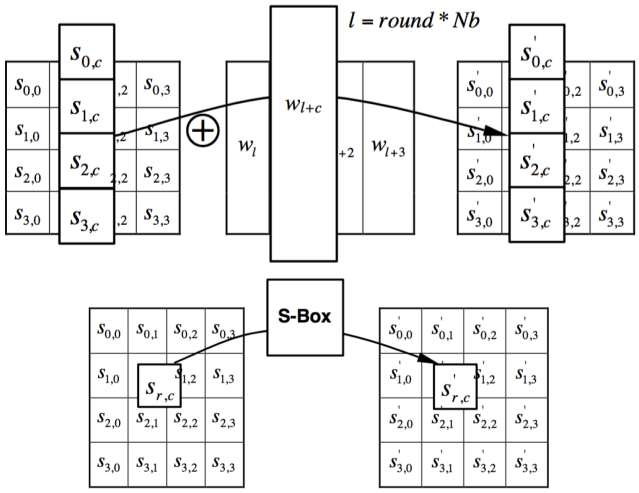


FIGURE 1.2: AddRoundKey (top) and SubBytes (bottom) operate over the State byte by byte, independently. Source: [NIS].

KeySchedule

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, for $i > 3$

$$\begin{cases} k_{a,b} = k_{a-4,b} \oplus k_{a-1,b} & \text{if } a \not\equiv 0 \bmod 4 \\ k_{a,b} = k_{a-4,b} \oplus \text{Sbox}(k_{a-1,(b+1)\bmod 4}) \oplus \text{Rcon}(a) & \text{if } a \equiv 0 \bmod 4, \end{cases}$$

where $Rcon(a) = \{02\}^{a-1}$ in the Rjindael finite field.[2]

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 $\oplus$.

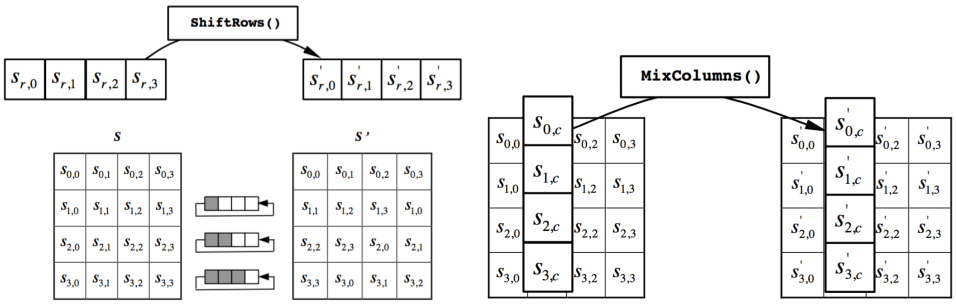


FIGURE 1.3: ShiftRows operates over the State rows. MixColumns operates over the State columns. Source: [NIS].

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 *S-Box*). The SubBytes is composed of the following two functions:

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

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

  where c_i is the i̧bit 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 bytes 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 $a(X) = \{03\}X^3 + \{01\}X^2 + \{01\}X + \{02\}$.

1.1.2 Description of RSA

The RSA cryptosystem, proposed in 1978 by Rivest, Shamir and Adleman [**rivest1978method**], represents the first practical realization of the use of trapdoor functions to provide asymmetric cryptography (a concept proposed two years earlier by Diffie, Helman and Merkle [**diffie1976new**]. RSA is one of the most deployed, analysed, challenged, and discussed cryptosystems of all times, the seminal paper [**rivest1978method**] counting today 18950 citations.[3]

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

[3] From Google Scholar, visited on January 2018.

1.2 Secured 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, impossible 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 asked to be delivered. The registrations of a first patent by Roland Moreno in 1974 and of a second one by Michel Ugon in 1977 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 secured component, which is typically an integrated circuit containing a 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, it seems 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 secured components, for example the *Trusted Execution Environment* (TEE), which as a secured part of the main processor of a smartphone or tablet, and the *Trusted Platform Module* (TPM), which is a secure element providing cryptographic functionalities to a motherboard.

1.2.1 Embedded Cryptography Vulnerabilities

Side-Channel Attacks

Until the middle of nineties, the security of embedded cryptosystems was considered as equivalent to the mathematical security of the cryptographic algorithm. In classical cryptanalysis an attacker has usually the knowledge of the algorithm (in accordance to the Kerckhoff's principle) and of some inputs and/or outputs. Starting from this 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 material component [Koc96]: an attacker can indeed inspect its component during the execution of the cryptographic algorithm, monitor some physical quantities, for example 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 variable (the so-called *sensitive variables* of the algorithm) is sufficient to retrieve the secret key. After these

TABLE 1.1: Classification of Attacks

Hardware Attacks	Passive	Active	
			Invasive
			Semi-Invisive
	SCAs	FAs	Non-Invasive
Software Attacks			

first works it was shown that other observable physical quantities contained *leakages* of 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 in a small-size architecture, such as smart cards were vulnerable to this kind of Side-Channel Attacks, this 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].

A Classification of the Attacks to Secured 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 attacks that can be performed to break cryptographic devices security claims. First of all software attacks and hardware attacks must be distinguished. Software attacks only exploit vulnerability coming from the way the component's code is written. Hardware attacks are still 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 is let 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 make the normal behaviour of the device change.

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

Semi-invasive attacks: as in invasive attacks the device is depackaged, 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 literature, the term Side-Channel Attacks commonly denotes the passive non-invasive attacks. In the same way, active non-invasive attacks are often referred to as *Fault Injection 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, which implies consequent great risks in case of vulnerabilities of such largely

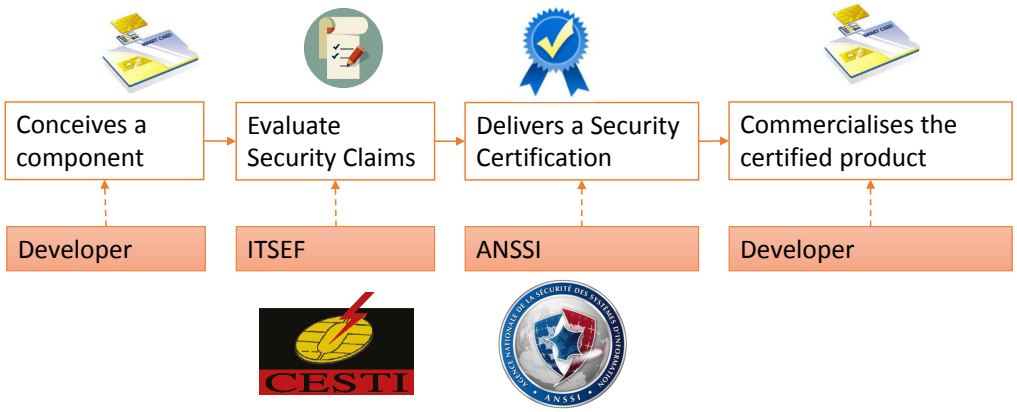


FIGURE 1.4: The actors of French Certification Scheme

diffused devices, and the existence of a wide range of attacks exploiting vulnerabilities coming from the way cryptography is embedded. These factors justify the importance and necessity to ensure reliability on the security claims of commercialised secured components and thus 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 stronger 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).

The actors

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

- **The Developer**, who conceives a product and wish sell it a certified secured product. He send 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. 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.

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

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) and then lists its *Security Functional Requirements* (SFR), choosing by those proposed by the CC. In practice, and to ease de 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* (PP), associated with the required SFRs; for example *smart card* or *TEE* designate some precise PPs.

Evaluation Assurance Level and Security Assurance Requirements

In CC seven *Evaluation Assurance Level* (EAL) are defined, and 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 in called the *attack* class. Each class is sub-divided in several *families* (excepted the attack 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 need 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.

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

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 efficacy of its the 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 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, listing the possibly dangerous attack path, basing on a code analysis, and on the state-of-the-art attacks list in general provided by working groups dedicated to the secured 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 base of several criteria. The goal of the cotation table is to provide a metric able to compare very different kind 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 grade, thus the testing plan is asked to be as complete as possible. The state-of-the-art of the attacks is periodically upgrades by the *JIL*[4] *Hardware Attacks Subgroup*, a subgroup of the working committee *Senior Officials Group Information Systems Security* (SOG-IS) which coordinate 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 perform the attack. The total score of an attack is the sum of scores assigned to each factor. To obtain the AVA_VAN5 grade every attack tested by the evaluators must have been rated at least 31.

The Evaluation Technical Rapport

The evaluation ends with the redaction by the evaluators of an *Evaluation Technical Rapport*, which is transmitted to the certification body. The last analyses the ETR and, in case 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 succeed attacks. If the component is certified it means that the score of such attacks was higher than 31, and such vulnerability are kept as *residual vulnerabilities*. The ETR will probably be read annually by the

[4] 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

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*, sometimes interpretable as *samples 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 certain variables, for example some random numbers used by cryptographic algorithm, of load specific software. The evaluator generally exploits this possibility to ease the task of characterization of the behaviour of the device.

In the context of Side-Channel Attacks, when such an characterization phase is possible we talk about *profiling attacks*. Due to the favourable condition of this attacks their are commonly considered the more dangerous, allowing a sort of worst-case security analysis. As we will see in next Chapters, this thesis is mainly focused over the 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 of the so-called *Points of Interest* (PoI), strictly link to the more general problem of dimensionality reduction. Indeed,

1.3.1 Foreword of this Thesis: Research of Points of Interest

1.3.2 Dimensionality Reduction Approach

Linear Methods for First-Order Attacks

Kernel Methods for Higher-Order Attacks

1.3.3 Neural Network Approach

Chapter 2

Introduction to Side-Channel Attacks

2.1 Introduction to Side-Channel Attacks

Side-Channel Attacks (SCA) belong to 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 for the fact that they are based on information gained from the physical implementation of a cryptosystem, rather than theoretical weaknesses in the algorithms.

Divide-and-Conquer

Side-Channel Attacks go beyond the cryptographic complexity coming from key size, and assuring in classical cryptanalysis a certain security level. Indeed, no matter the size of a cryptographic algorithm inputs, outputs, keys and internal variables, once it is implemented in hardware operations will always be executed over variables of a bounded size. Such a bound depends on the hardware architecture. For example, in an 8-bit architecture an RSA with 1024-bit-sized key, modulo and plaintext will be somehow implemented as multiple operations over 8-bit blocks of observable data. This fact allows an attacker to apply the *divide-and-conquer* strategy: if his goal is to retrieve the full 128-bit AES key or 1024-bit RSA key, he will smartly divide his problem in retrieving small parts of such keys at time, called *subkeys*.

Sensitive Variable

In order to make inference on such subkeys, an attacker acquires side-channels signals, *e.g.* instantaneous power consumption or electromagnetic irradiation, during the execution of the algorithm. Such signals are collected into vectors called *traces* (or *acquisitions*). Such traces will be denoted as observations $\mathbf{x}_i$ 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 finding the right association between a trace (or a set of traces) and the value assumed by a target *sensitive variable* Z during its/their acquisition. A sensitive variable is a quantity handled during the algorithm 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 exploits

always the same key for a given embedded primitive. When the target is not variable we are performing a *simple attack* (see below);

- a cryptographic variable that depends over a sufficiently small subkey and a part of a known input variable E: $Z = f(K, E)$ - this is the 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 HW represent the Hamming weight. We will see later in which sense it can be interesting to not target a variable but a non-injective function of a variable as the Hamming weight is;
- an operation (ex: $Z \in \{\mathrm{square}, \mathrm{multiply}\}$)
- a register (ex: Z is the register used to store results of 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 permit an attacker to make inference about a secret of the implemented algorithm.

Leakage Models

The underlying hypothesis of a SCA is that some information about internal variables (or parts of internal variables) of the implemented algorithm leak during its execution through some observable *side* channels. Such leakages are collectable in the form of signal traces, observing such channels. 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 are needed, for example no noise is taken into account. As explained in 1.2.1, 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 d-th order masking schemes (cf. 2.2.3).
The most common passive leaking channel considered in literature being 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 modelisation, 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, that 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 modelise 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*. In this model the leakage is a random variable obtained by the sum of a deterministic function of the sensitive variable Z and a random noise. In general the noise has a Gaussian distribution of null mean and variance σ^2:

$$X = \phi(Z) + B\,, \tag{2.1}$$

with $B \approx \mathcal{N}(0, \sigma^2)$.

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 the same sensitive variables are involved in many internal operations during the overall algorithm (for example as in [Bat+16]). Algebraic SCAs (see below) are horizontal attacks as well, in which inferences over potentially every cryptographic variable are done, followed by a deep analysis of algebraic relations between such sensitive variables. Collision attacks (see below) may (or not) be executed in a horizontal fashion. 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 both exploit vertical and horizontal leakages.

Profiling vs Non-Profiling SCAs

As anticipated in Sec. 1.3, when an open sample of the attacked device is available to make a prior characterization 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 thus divided into two distinct phases. The first one, called *profiling phase* of *characterization* phase exploits some 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 $(\mathbf{x}_i, z_i)_{i=1,\dots,N_p}$ 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, unknown secret key, and take advantage of the previous characterization to infer over it.
As we will see in Chapter 3, in machine learning domain the analogous of profiling attacks context is known under the name of *supervised machine learning*. In supervised machine learning couples $(\mathbf{x}_i, z_i)_{i=1,\dots,N_p}$ are available and are called *training examples*. The profiling phase is referred to as *training* of *learning* and the attack phase is assimilable to the so-called *test phase*. If no example is available we talk about *unsupervised machine learning*, that we can consider analogous to the non-profiling SCAs branch.

Classes

Throughout this thesis, and each time a profiling attack scenario is supposed, we will refer to elements of $\mathcal{Z}$ as *classes*. We will say that acquired traces associated to a same value $z \in \mathcal{Z}$ *belong* to the same class z. Eventually we will use the term *label* to denote the classes, in such a way that each trace is *labelled* by an element of $\mathcal{Z}$, or associated to a label $z \in \mathcal{Z}$.

Simple vs Advanced SCAs

A simple attack is an attack that only need one trace to be applied, except for profiling acquisitions. 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, and 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.* proposed for example a simple power analysis (SPA) strategy to retrieve the whole AES key observing leakages from a single execution of the AES key expansion [Man02]. When Algebraic SCAs (see below) appeared in literature in 2009, their aim was to be a strategy to perform simple attacks as well. Attacks for which many observations are acquired with fixed entry parameters and by consequence in which the observed leakage always corresponds to the same value of Z are still considered simple attacks. Generically speaking, the attacker exploits the several acquisitions in mainly two ways: he computes their average before performing the classification 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) to guess the right label. We observe that this approach with several observation allows the attacker to reduce the noise impact, while observing many times the same amount of information.
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.

Template Attack

Introduced in 2004 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, statistic 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

$(\mathbf{x}_i, z_i)_{i=1,\dots,N_p}$ and exploit them to estimate the class-conditional densities

$$p_{\vec{X}}(\mathbf{x}_i \,\big|\, Z = z) \,, \tag{2.2}$$

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

$$p_Z(z \,\big|\, \vec{X}) = \frac{p_{\vec{X}}(\mathbf{x}_i \,\big|\, Z = z) p_Z(z)}{p_{\vec{X}}(\mathbf{x}_i)} \,. \tag{2.3}$$

In the attack phase the attacker acquires new traces that he only can associate to the public parameter E, obtaining couples $(\mathbf{x}_i, e_i)_{i=1,\dots,N_a}$. 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 a score given by the joint a-posteriori probability that follows, exploiting distributions (2.2), (2.3):

$$d_k = \prod_{i=1}^{N_a} \mathrm{Prob}[Z = f(e_i, k) \,\big|\, \vec{X} = \mathbf{x}_i] \,. \tag{2.4}$$

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

$$\hat{k} = \underset{k}{\mathrm{argmax}}\, d_k \,. \tag{2.5}$$

Remark. Since the marginal probability density $p_{\vec{X}}(\mathbf{x}_i)$ of (2.3) does not depend on key hypothesis, it is usually neglected. Moreover, in many cases the Z follows a uniform distribution, so its probability mass function $p_Z(z)$ appearing in (2.3) does not influence the ranking of key hypothesis, then it is often neglected as well.

Remark. 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 (2.5) is known as *Bayes (optimal) classifier*.[1] It is proven to be the optimal choice to reduce de misclassification error [Bis06].

In general this approach theoretically exploits all available information and is optimal under an information-theoretical point of view. The crucial point is the estimation of the class-conditional densities (2.2): 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 2.2 is the one applied in TA [CRR03]: it consists in making a class-conditional multivariate Gaussian distribution assumption

$$\vec{X} \,\big|\, Z = z \approx \mathcal{N}(\boldsymbol{\mu}_z, \Sigma_z) \,, \tag{2.6}$$

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

[1] 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. 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 (2.3), 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. The pooled estimation gains in accuracy.

Remark. 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 5.

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

Points of Interest and Dimensionality Reduction

The 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 these researches is to study and develop techniques to characterize PoIs and to apply conve- nient 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. Considering 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 $\epsilon\colon \mathbb{R}^D \to \mathbb{R}^C$, called *extractor* in this paper.

Algebraic Side-Channel Attacks

Algebraic SCAs (ASCA) were firstly proposed in 2009 [RS09; RSVC09]. Their aim was to combine profiling SCA strategies (templates-like) to classical cryptanalysis techniques: in these first 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 beginning or lasting rounds, were the cryptographic diffusion is limited. This optimal exploitation of the information, allows the ASCA strategy provide efficient simple attacks, *i.e.* succeeding with a single attack trace.
Two main weaknesses of this approach are pointed out. 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 in 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 approach 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 literature about this research axe.

SCA Metrics

2.2 Main Side-Channel Countermeasures

2.2.1 Random Delays and Jitter

2.2.2 Shuffling

2.2.3 Masking

2.3 Higher-Order Attacks

2.3.1 Higher-Order Moments Analysis and Combining Functions

2.3.2 Profiling Higher-Order Attacks

Profiling with Masks Knowledge

Profiling without Masks Knowledge

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 [**Mitchell1997**]: " 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 characterized 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 an algorithm that learns from data, in the sense that is an algorithm able to improve a computer program's performance at some task via an experience.

3.1.1 The Task, the Performance and the Experience

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 $\mathbf{x} \in \mathbb{R}^D$, which is in turn a collection of *features* $\mathbf{x}[i]$. In SCA context an example is a side-channel trace, which is in turn a collection of time samples, that are 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 $f : \mathbb{R}^D \to \mathbb{R}$.
- *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 $f : \mathbb{R}^D \mathcal{Z}$. 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. As slightly variant solution to the classification task consists in constructing a function $f : \mathbb{R}^D \to \{0,1\}^{|\mathcal{Z}|}$, once have assigned to each class z^j a *one-hot encoding*, *i.e.* a $|\mathcal{Z}|$-dimensional vector, with all entries equal to 0 and the j-th entry equal to 1: $z^j \to \vec{z}^j = (0,\ldots,0,\underbrace{1}_{j},0,\ldots,0)$. A variant of the classification task consists in finding a function $\hat{f}$ 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 state if two hand-written

signature have been produced by the same person. The learning algorithm is thus asked to construct a function $f : \mathbb{R}^D \times \mathbb{R}^{traceLength} \rightarrow \{0, 1\}$. A variant of such a task consists in finding for a pair of inputs the probability distribution for them being instance or not of a same class. This problem differs from the classification one essentially for the for of the input.

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

The performance P designs a quantitative measure of the abilities 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 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 differences raised to the square between the correct output and the one predicted by the model.
One of the crucial challenge 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 E denotes the typology of access to data and information the algorithm is allowed during learning. In this context we principally distinguish two family of learning algorithms:

- the *supervised* learning algorithms experience a dataset of examples, each associated in general to a *target* or *label*. The term supervised reflects the fact the the learning is somehow guided by an instruct that knows the right answer over the learning dataset;
- the *unsupervised* learning algorithms experience a dataset, without any associated target. They tries 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*. The Principal Component Analysis, that will be discussed in Chapter 5 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 how for SCA context a supervised version of 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. Let us introduce a linear regression model so tackle the regression task: we want to construct a linear function $f : \mathbb{R}^D \rightarrow \mathbb{R}$, that takes an input $\mathbf{x}$ and outputs $y = \mathbf{w}^\intercal \mathbf{x}$,

where $\mathbf{w} \in \mathbb{R}^D$ is a vector of *parameters* that have to be learned by a learning algorithm.[1] We want this model well describe some data and we suppose have two available datasets of such data: $\mathcal{D}_{\text{train}} = (\vec{\mathcal{X}}_{\text{train}}, \mathcal{Y}_{\text{train}})$, to let the learning algorithm experience on, and $\mathcal{D}_{\text{test}} = (\vec{\mathcal{X}}_{\text{test}}, \mathcal{Y}_{\text{test}})$ to evaluate the performance of the obtained model over some unseen data. Let us choose the MSE over the test set to evaluate such performances. If we collect the, let say N, examples of a dataset as column of a measure matrix $\mathbf{M} \in \mathbb{R}^{D \times N}$ and the related target into a vector $\mathbf{y} \in \mathbb{R}^N$, and let the learned model predict targets y by outputting $\hat{y} = \mathbf{wx}$, then the MSE is given by

$$\text{MSE}_{\text{test}} = \frac{1}{m} \left\| \hat{\mathbf{y}}_{\text{test}} - \mathbf{y}_{\text{test}} \right\|_2^2 . \tag{3.1}$$

The MSE is the performance measure in this example, meaning that we consider the model performs well the most such an MSE is small. So the goal of the learning algorithm is somehow minimize the MSE_{test}. But the learning algorithm only experiences on the $\mathcal{D}_{\text{train}}$ dataset. An intuitive way to act, that can be proven be the maximum likelihood solution to the problem, is to minimize $\text{MSE}_{\text{train}}$ by solving an easy optimization problem. The solution to such a optimization problem can be given in closed form, by means of the pseudo-inverse matrix of $\mathbf{M}_{\text{train}}$:

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

$\mathcal{D}_{\text{train}}$ $\mathcal{D}_{\text{test}}$ $\mathcal{D}_{\text{profiling}} = (\vec{\mathcal{X}}, \mathcal{Y})$

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.

Classify means assigning to an example $\mathbf{x} \in \mathbb{R}^D$ a label $z \in \mathcal{Z}$, 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. Data sets 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, or a discriminative model is constructed. For this example we consider a probabilistic approach, constructing a generative model. This example will allow to introduce some interesting functions, such as the *logistic sigmoid* and the *softmax*, that will play a role in the construction of neural network (see Chapter 7, and to show how adding some assumptions on the data distributions one can justify, in contexts where the goal of the learning algorithm is making decisions directly, dispensing with any probabilistic interpretation, the exploitation of linear discriminant functions.

Construct a generative probabilistic model implies modelling the class-conditional probabilities $p(\mathbf{x} \mid z^j)$ for $j \in [1, \ldots, |\mathcal{Z}|]$ as well as the class priors $p(z^i)$. Let us first

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

consider in a 2-class context, *i.e.* $|\mathcal{Z}| = 2$. Then the posterior probability for the class z^1 is the following:

$$p(z^1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid z^1)p(z^1)}{p(\mathbf{x} \mid z^1)p(z^1) + p(\mathbf{x} \mid z^2)p(z^2)} \,. \tag{3.3}$$

To compare the two classes, we can look to their *log-likelihood ratio*:

$$a = \log\left[\frac{p(z^1 \mid \mathbf{x})}{p(z^2 \mid \mathbf{x})}\right] = \log\left[\frac{p(\mathbf{x} \mid z^1)p(z^1)}{p(\mathbf{x} \mid z^2)p(z^2)}\right] . \tag{3.4}$$

Then the discriminative criteria can be assigning to $\mathbf{x}$ the class z^1 iff $a > 0$. The decision boundary is given by the surface defined by $p(\mathbf{x} \mid z^1)p(z^1) = p(\mathbf{x} \mid z^2)p(z^2)$. We remark that we Eq. (3.3) rewrites as

$$p(z^1 \mid \mathbf{x}) = \frac{1}{1 + e^{-a}} = \sigma(a) \,, \tag{3.5}$$

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

$$p(z^j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid z^j)p(z^1)}{\sum_k p(\mathbf{x} \mid z^k)p(z^k)} = s(\mathbf{a})[k] \,, \tag{3.6}$$

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

$$\mathbf{a}[j] = \log\left[p(\mathbf{x} \mid z^j)p(z^j)\right] \,, \tag{3.7}$$

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

$$s(\mathbf{a})[k] = \frac{e^{\mathbf{a}[k]}}{\sum_{j=1}^{|\mathcal{Z}|} e^{\mathbf{a}[j]}} \,. \tag{3.8}$$

Let us now introduce two assumptions about class-conditional densities: we will suppose they follow a Gaussian distribution with parameters μ_j, Σ_j, and that all class-conditional densities share the same covariance matrix $\Sigma_j = \Sigma$, so that

$$p(\mathbf{x} \mid z^j) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_j)^\intercal \Sigma^{-1}(\mathbf{x}-\mu_j)} \,. \tag{3.9}$$

Under this assumptions Eq. (3.4) rewrites as:

$$a = \log(\frac{p(z^1)}{p(z^2)}) - \frac{1}{2}\mu_1^\intercal\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^\intercal\Sigma^{-1}\mu_2 - \mathbf{x}^\intercal\Sigma^{-1}(\mu_2 - \mu_1) = \mathbf{w}^\intercal\mathbf{x} + w_0, \tag{3.10}$$

where we set

$$\begin{aligned} \mathbf{w} &= \Sigma^{-1}(\mu_1 - \mu_2) \\ w_0 &= \log\frac{p(z^1)}{p(z^2)} - \frac{1}{2}\mu_1^\intercal\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^\intercal\Sigma^{-1}\mu_2. \end{aligned}$$

The quadratic terms in $\mathbf{x}$ in the exponent of the Gaussian density have cancelled thanks to the common variance assumption, 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.

[2] 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[3] one can see the term $\mathbf{w}^\intercal\mathbf{x}$ of (3.10) as a projection of the input $\mathbf{x}$ onto a one-dimensional subspace of $\mathbb{R}^D$ orthogonal to the decision boundary mentioned above, then classify the obtained dimensionality-reduced examples by the means of a 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 5, to which we will refer to LDA dimensionality reduction by a common abuse, is equivalent to the dimensionality reduction obtained in this example, making the Gaussian assumption over the class-conditional probabilities, equipped of a common covariance matrix assumption.
Relaxing the assumption of a shared covariance matrix and allowing each class-conditional density $p(\mathbf{x} \mid z^j)$ to have its own covariance matrix Σ_j, then the earlier cancellations will no longer occur, and the discriminant a turns out to be a quadratic function of $\mathbf{x}$. This gives rise to the so-called *Quadratic Discriminant Analysis*, that we already mentioned in Chapter 2 for its analogy with Template Attacks.

The two assumptions leading to the log-likelihood ratio (3.10), also leads to the following expression for the posterior probability for z^1, directly implied by (3.5):

$$p(z^1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\intercal\mathbf{x} + w_0)\ . \tag{3.11}$$

Thus such a posterior probability is given by the sigmoid acting to a linear function of $\mathbf{x}$. 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 $\mathbf{x}$. This kind of *generalized linear model* can be thus used in a probabilistic discriminant approach, where the posterior conditional probabilities are directly modelised 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 7 when considering neural networks as models.

3.1.4 Training, Validation and Test Sets

3.1.5 Capacity, Underfitting, Overfitting and Regularization

3.1.6 No Free Lunch Theorem

3.2 Machine Learning Applications in Side-Channel Context

3.2.1 Profiled Attack as a Classification Problem

remark that LDA is first of all a linear method for classification and has been introduced in SCA many years ago as preprocessing for Gaussian TA

Support Vector Machine

Random Forest

Neural Networks

[2] An analogous result can be obtained in the multi-class problem.
[3] again extensible to the multi-class case

Part II

Contributions

Chapter 4

Points of Interest

4.1 Motivations

4.1.1 The Curse of Dimensionality

4.2 Selection on Points of Interest: Classical Statistics

4.3 Related Issues: Leakage Detection and Leakage Assessment

test statistici agli ordini superiori, soprattutto per masking scheme hardware, CHES 2017 e precedenti

4.4 Generalized SNR for Multi-Variate Attacks

4.5 Observations Leading to Take a Dimensionality Reduction Approach

Chapter 5

Linear Dimensionality Reduction

5.1 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 Zoubin [CG15]. They proposed a generalized optimization framework for all linear dimensionality techniques, survey a dozen different techniques and mention some important extension such as kernel mappings.

Among the surveyed methods in [CG15] we find the two mainly considered in SCA literature: the Principal Components 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, 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 parts of the data and randomly setting the projecting coefficient, and by tracking the improvements (or lack thereof) of the projection when modifying it 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 naturally fits to the problem of higher-order SCA. Thus this technique appears advantageous in higher-order context, where it is used as a PoI selection tool. It is its version for the first-order attacks which turns out in a method to linearly reduce dimensionality. Nevertheless, in this context it 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 6.

In SCA literature, one of the open issues in PCA concerns the choice of the 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 this chapter the classical PCA technique is first of all described (Sec. 5.2) . In Sec. 5.3, we recall the previous applications of PCA in SCA context, highlighting the difference between its unsupervised and supervised declination. Then we propose our contribution to the choice of components open issue: our solution is based on the Explained Local Variance (ELV) notion, that we will define and discuss on 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 this paper, 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 LDA technique will be described in Sec. 5.4, together with (in Sec. 5.5 its previous uses in SCA literature, the description of the SSS problem and some solutions coming from the Pattern and Face Recognition communities [BHK97; Che+00; Hua+02; YY01]. Through some experiments depicted in Sec. 5.6 we will conclude about the effectiveness of the PCA-ELV solution. Finally, in Sec. 5.7 we will experimentally argue about the non robustness of these techniques to data misalignment.

5.2 Principal Component Analysis

The Principal Component Analysis (PCA) is a technique for data dimensionality reduction. The same 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

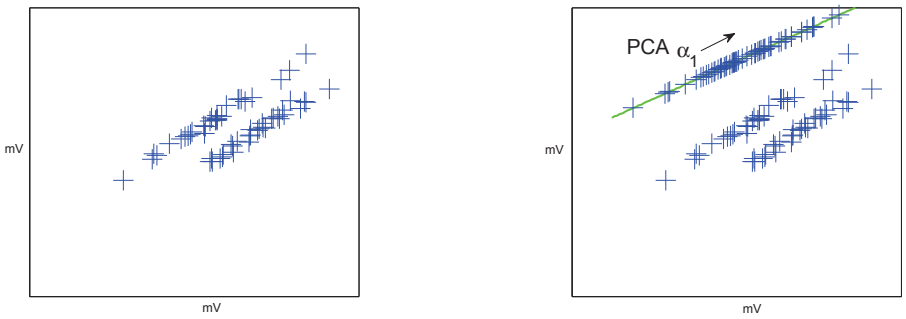


FIGURE 5.1: PCA: some 2-dimensional data projected into their 1-dimensional principal subspace.

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. In Appendix **??** the equivalence between the two approaches is depicted. An example of 2-dimensional data projected over their 1-dimensional principal subspace is depicted in Fig. 5.2

Let $(\mathbf{x})_{i=1..N}$ 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

$$\mathbf{S} = \frac{1}{N}\mathbf{M}\mathbf{M}^{\intercal} . \tag{5.1}$$

Let us assume for the moment that we have 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 $Y_1 = \boldsymbol{\alpha}_1\vec{X}$, i.e. projecting data over a single $1 \times D$ vector $\boldsymbol{\alpha}_1$, in such a way the the variance of the obtained data is maximal. The vector $\boldsymbol{\alpha}_1$ 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 $\boldsymbol{\alpha}_i$, while we will refer to the variable $Y_j = \boldsymbol{\alpha}_j\vec{X}$ as *i-th Principal Variable (PV)*. Realizations of the PVs are given by the measured data projected over the j-th PC, for example we have N realizations of Y_1:

$$y_1[i] = \boldsymbol{\alpha}_1\mathbf{x} \text{ for } i = 1, \ldots, N . \tag{5.2}$$

Let us collect these realizations in a vector $\mathbf{y}_1 = \boldsymbol{\alpha}_1\mathbf{M}$; 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

$$\frac{1}{N}\mathbf{y}_1\mathbf{y}_1^{\intercal} = \frac{1}{N}\boldsymbol{\alpha}_1\mathbf{M}\mathbf{M}^{\intercal}\boldsymbol{\alpha}_1^{\intercal} = \boldsymbol{\alpha}_1\mathbf{S}\boldsymbol{\alpha}_1^{\intercal} . \tag{5.3}$$

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

Obviously, we can attend a value for the variance as high as wished, raising the modulo $\|\boldsymbol{\alpha}_1\| = \sqrt{\boldsymbol{\alpha}_1^T \boldsymbol{\alpha}_1}$. In order to let the maximization problem have a solution, we impose a restriction over it: $\boldsymbol{\alpha}_1^T \boldsymbol{\alpha}_1 = 1$.
Let us handle this constrained optimization problem making use of Lagrange multipliers:

$$\Lambda(\boldsymbol{\alpha}_1, \lambda) = \boldsymbol{\alpha}_1 \mathbf{S} \boldsymbol{\alpha}_1^\intercal - \lambda(\boldsymbol{\alpha}_1 \boldsymbol{\alpha}_1^\intercal - 1) \tag{5.4}$$

and let us compute de partial derivative of Λ with respect to $\boldsymbol{\alpha}_1$

$$\frac{\partial \Lambda}{\partial \boldsymbol{\alpha}_1} = 2\mathbf{S}\boldsymbol{\alpha}_1^\intercal - 2\lambda \boldsymbol{\alpha}_1^\intercal \,. \tag{5.5}$$

Thus, stationary points of Λ verify

$$\mathbf{S}\boldsymbol{\alpha}_1^\intercal = \lambda \boldsymbol{\alpha}_1^\intercal \,, \tag{5.6}$$

which implies that $\boldsymbol{\alpha}_1^\intercal$ must be an eigenvector of $\mathbf{S}$, with λ its correspondent eigenvalue. Multiplying both sides of Eq. (5.6) by $\boldsymbol{\alpha}_1$ on the left, we remark that

$$\boldsymbol{\alpha}_1 \mathbf{S} \boldsymbol{\alpha}_1^\intercal = \lambda \boldsymbol{\alpha}_1 \boldsymbol{\alpha}_1^\intercal = \lambda, \tag{5.7}$$

which means that the variance of the obtained variable $\mathbf{y}_1$ equals λ. For this reason $\boldsymbol{\alpha}_1$ must be the leading eigenvector of $\mathbf{S}$.

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 $\mathbf{Y} = [\frac{\boldsymbol{\alpha}_1}{\boldsymbol{\alpha}_2}]\vec{X}$ 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 $\mathbf{Y}$, which is defined as the determinant of its covariance matrix, instead of its trace.

Let us write, as in previous case, the Lagrangian of the problem

$$\Lambda = \boldsymbol{\alpha}_1 \mathbf{S} \boldsymbol{\alpha}_1^\intercal + \boldsymbol{\alpha}_2 \mathbf{S} \boldsymbol{\alpha}_2^\intercal - \lambda_1(\boldsymbol{\alpha}_1 \boldsymbol{\alpha}_1^\intercal - 1) - \lambda_2(\boldsymbol{\alpha}_2 \boldsymbol{\alpha}_2^\intercal - 1) \,. \tag{5.8}$$

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

$$\mathbf{S}\boldsymbol{\alpha}_1^\intercal = \lambda_1 \boldsymbol{\alpha}_1^\intercal \tag{5.9}$$

$$\mathbf{S}\boldsymbol{\alpha}_2^\intercal = \lambda_2 \boldsymbol{\alpha}_2^\intercal \,, \tag{5.10}$$

which means that $\boldsymbol{\alpha}_1^\intercal$ and $\boldsymbol{\alpha}_2^\intercal$ 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 we want to maximize the sum of such variables we have to choose $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ as the two leading vectors of $\mathbf{S}$.
Let us remark that the covariance between Y_1 and Y_2 is given by $\boldsymbol{\alpha}_1^T \mathbf{S} \boldsymbol{\alpha}_2$ which equals zero, since $\boldsymbol{\alpha}_1^T \boldsymbol{\alpha}_2 = 0$, by orthogonality. In particular the principal variables are uncorrelated, which is a remarkable property of the PCA.

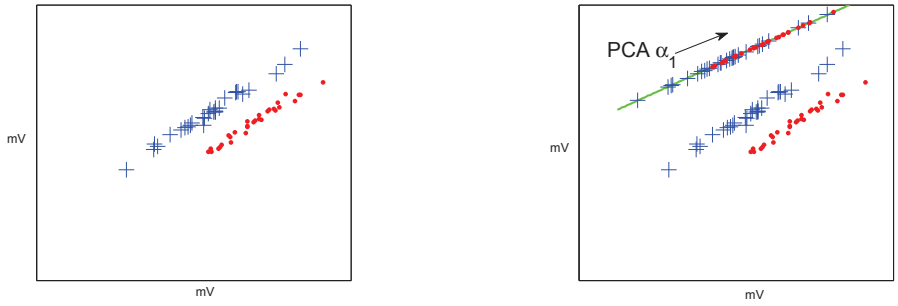


FIGURE 5.2: PCA: some 2-dimensional data projected into their 1-dimensional principal subspace.

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 $\mathbf{S}$.

5.3 Application of PCA in SCAs

5.3.1 Original vs Class-Oriented PCA

The classical version of PCA method is unsupervised, in the sense that is does not take into account the information about the value assumed by the target variable during the acquisition of data. On the other hand a profiling attacker is provided not only with a set of data $(\mathbf{x})_{i=1..N}$, but he can group such traces depending on the target value: this let him obtain a set of traces $(\mathbf{x})_{i=1..N}$ for each value of the target variable Z.
In this context, and for the sake of distinguish the target value assumed by the variable Z in new executions, the idea of the *Class-Oriented PCA is to consider* equivalent *all traces coming from each group, i.e. obtained by a same characteristic form plus a random noise. To estimate the characteristic form the sample means of the traces of each group is computed, obtaining* $\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1..N}\boldsymbol{x}$. *These sample means will take the place of data in PCA: we collect them as rows in the data matrix* $\boldsymbol{M}$, *shift the empirical mean of each component to zero, compute the covariance matrix* $\boldsymbol{S}$ *et select the leading* C *eigenvectors of* $\boldsymbol{S}$ *as optimal projecting directions.*
In this way we focus the attention on information that discriminate classes, i.e. target values.

5.3.2 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 important 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 now, an answer to the questions above has been given in [CK14b], linked to the concept of explained variance *(or* explained global variance, *EGV for short) of a PC* $\boldsymbol{\alpha}_i$*:*

$$\mathrm{EGV}(\boldsymbol{\alpha}_i) = \frac{\lambda_i}{\sum_{k=1}^{r}\lambda_k}\,, \tag{5.11}$$

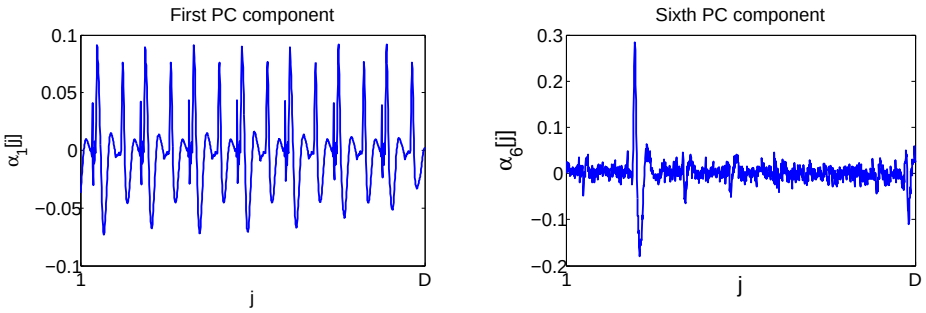


FIGURE 5.3: First and sixth PCs in DPA contest v4 trace set (between time samples 198001 and 199000)

where r is the rank of the covariance matrix $\mathbf{S}$, and λ_j is the eigenvalue associated to the j-th PC $\boldsymbol{\alpha}_j$. $\mathrm{EGV}(\boldsymbol{\alpha}_i)$ 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; that is why 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*

$$EGV(\boldsymbol{\alpha}_1) + EGV(\boldsymbol{\alpha}_2) + \cdots + EGV(\boldsymbol{\alpha}_C) \geq \beta \,. \tag{5.12}$$

However, if the adversary 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. 5.3 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.

To the best of our knowledge, a single method adapted to SCA context has been proposed until now to automatically choose PCs [Mav+12] while dealing with the issue raised in Fig. 5.3. It is based on the following assumption:

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

In the rest of the paper, we conduct our own analyses under Assumption 1 that we think to be 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 for example [***guhr1998random***]). They propose to use such a score to evaluate the eigenvectors* localization. *It is defined as follows:*

$$\mathrm{IPR}(\boldsymbol{\alpha}_i) = \sum_{j=1}^{D} \boldsymbol{\alpha}_i[j]^4 \,. \tag{5.13}$$

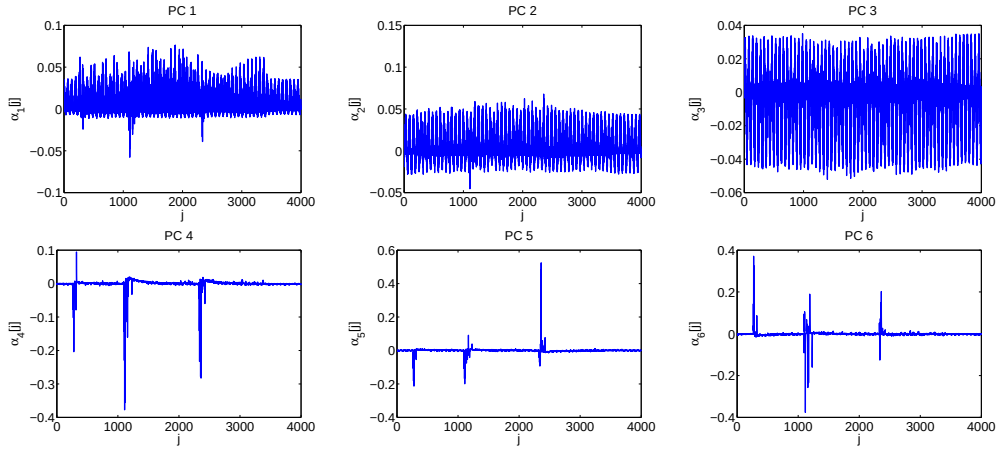


FIGURE 5.4: The first six PCs. Acquisition campaign on an 8-bit AVR Atmega328P (see Sec. 5.6).

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. One of the contributions of the present paper is to propose a new selection method, 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.*

5.3.3 The Explained Local Variance Selection Method

Riprendere le notazioni e mettere apposto i newcommand

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

$$\mathrm{ELV}(\boldsymbol{\alpha}_i, j) = \frac{\lambda_i \boldsymbol{\alpha}_i[j]^2}{\sum_{k=1}^{r} \lambda_k} = \mathrm{EGV}(\boldsymbol{\alpha}_i)\boldsymbol{\alpha}_i[j]^2 \,. \tag{5.14}$$

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 $\mathrm{ELV}(\boldsymbol{\alpha}_i, j_1^i) \geq \mathrm{ELV}(\boldsymbol{\alpha}_i, j_2^i) \geq \cdots \geq \mathrm{ELV}(\boldsymbol{\alpha}_i, j_D^i)$. It may be observed that the sum over all the $\mathrm{ELV}(\boldsymbol{\alpha}_i, j)$, for $j \in [1, \ldots, D]$, equals $\mathrm{EGV}(\boldsymbol{\alpha}_i)$. 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 $\boldsymbol{\alpha}_i$ to achieve its EGV. As we can see in Fig. **??***, 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 $\mathrm{ELV}(\boldsymbol{\alpha}_4, j_k^4)$, with $k \in [1, \ldots, 30]$, almost equals $\mathrm{EGV}(\boldsymbol{\alpha}_4)$, whereas the same sum for $i = 1$ only achieves about the 15% of $\mathrm{EGV}(\boldsymbol{\alpha}_1)$. 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 form of such components (Fig. 5.4): the leading ones are very 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. 5.6 (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.*

5.4 Linear Discriminant Analysis

Linear Discriminant Analysis

5.5 Application of LDA in SCAs

5.5.1 The Small Sample Size problem

5.6 Experimental Results

In this section we compare the different extractors 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 scenario. In the following only results of the two first is reported, the interested reader might refer to AppendixA for results of in the two other scenarios.

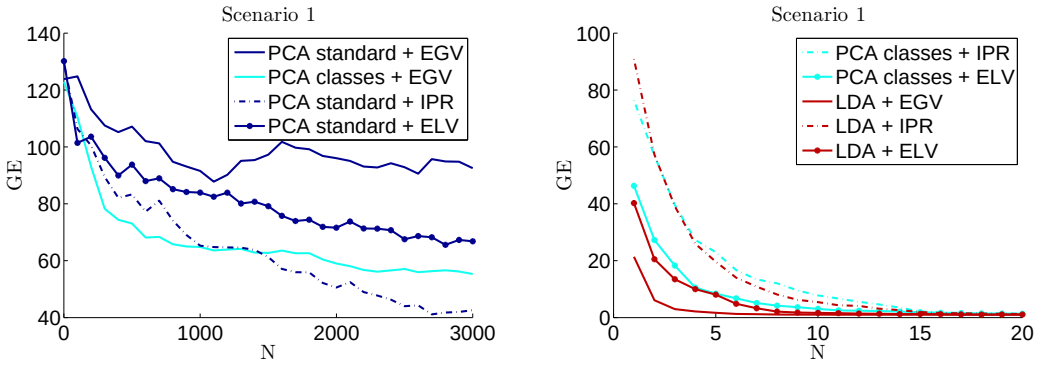
Scenario 1 *varying parameter: number of attack traces N_a,*

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* $\sharp$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. **??** *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. Consequently, 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. 5.3.2.*



(a) (b)

FIGURE 5.5: 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.

This part will maybe be useless: somewhere I will have described all trace sets

The testing adversary.

Scenario 1.

cos'e N_z *To analyse the dependence between the extraction methods presented in Sections 5.2 and 5.4 and the number of attack traces* N_a *needed to achieve a given GE, we fixed the other parameters as follows:* $N_z = 50$ *(*$N_p = 50 \times 256$*),* $C = 3$ *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. 5.5(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.*

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.* **??***, associated to either the standard selection, to which we abusively refer as EGV,*[1] *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* $N_z \geq 16$ *there is no need to approximate the LDA technique, so the classical one is performed. Results for this scenario are shown in Fig. 5.6. It may be noticed that the combinations class-oriented PCA + ELV/IPR select exactly the same components, for our data, see Fig. 5.6(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.5.6(d). The Direct LDA (Fig. 5.6(b)) method also provides a good alternative, while the other tested methods do not show a stable behaviour. The results in*

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

absence of the SSS problem (Fig.5.6(f)) 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.

5.7 Misaligning Effects

give parameters: 6 4 citare Choudary, Template Attacks over different devices *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. 5.6, and artificially misalign them through the technique proposed in Appendix A.2 with parameters* ***parameters here****. Then we tried to pre-process attack them through the 9 methodology tested in Scenario 1. It may be noticed in Fig. 5.7 that none of the 9 techniques is still efficient, included the optimal LDA+EGV that lead to null guessing entropy the synchronized traces using less 7 attack traces. In this case it cannot lead to successful attack in less than 3000 traces.*

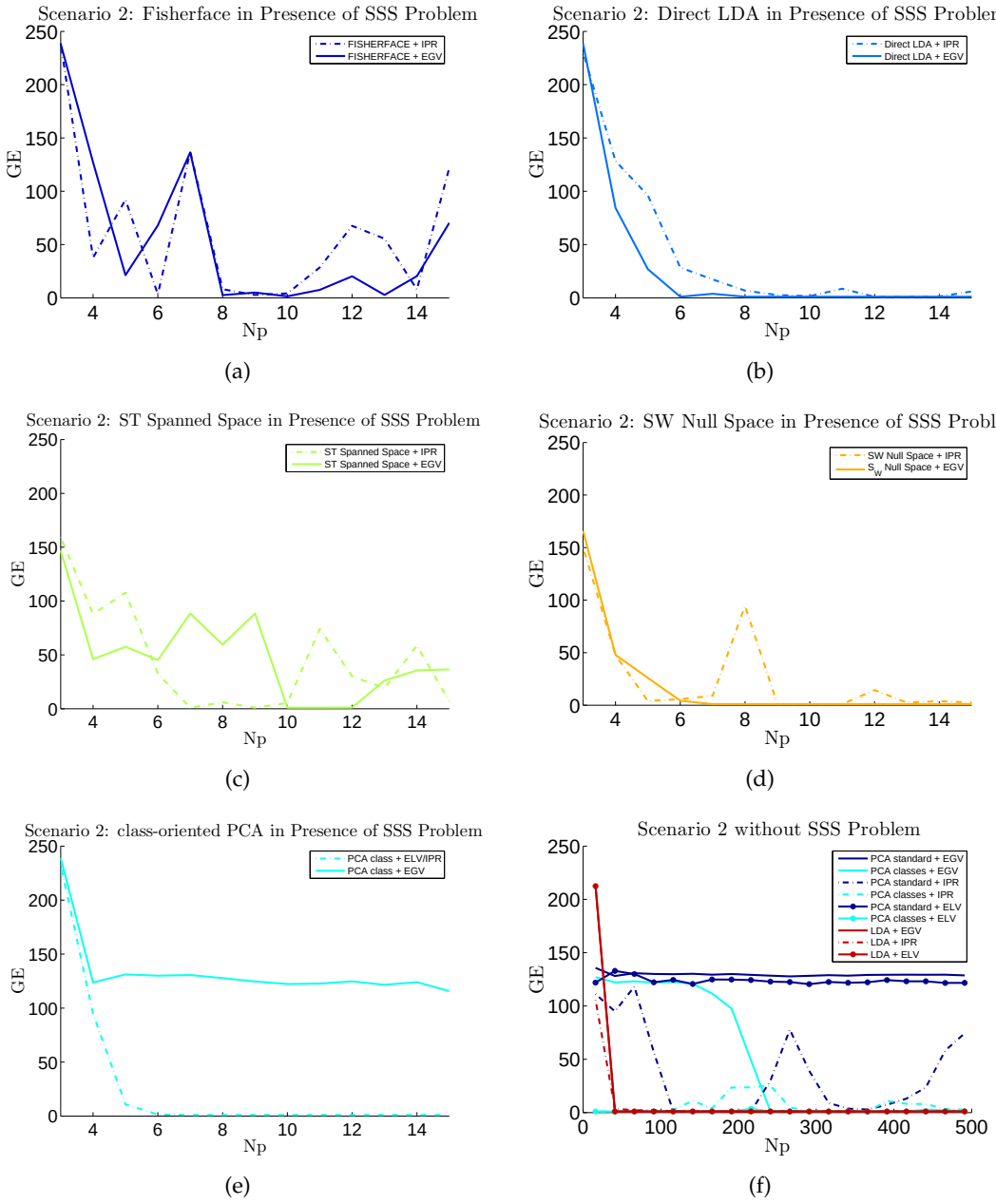


FIGURE 5.6: Guessing Entropy as function of the number of profiling traces. Figures (a)-(d): methods extending the LDA in presence of SSS problem; Figure (e): class-oriented PCA in presence of the SSS problem; Figure (f): number of profiling traces high enough to avoid the SSS problem.

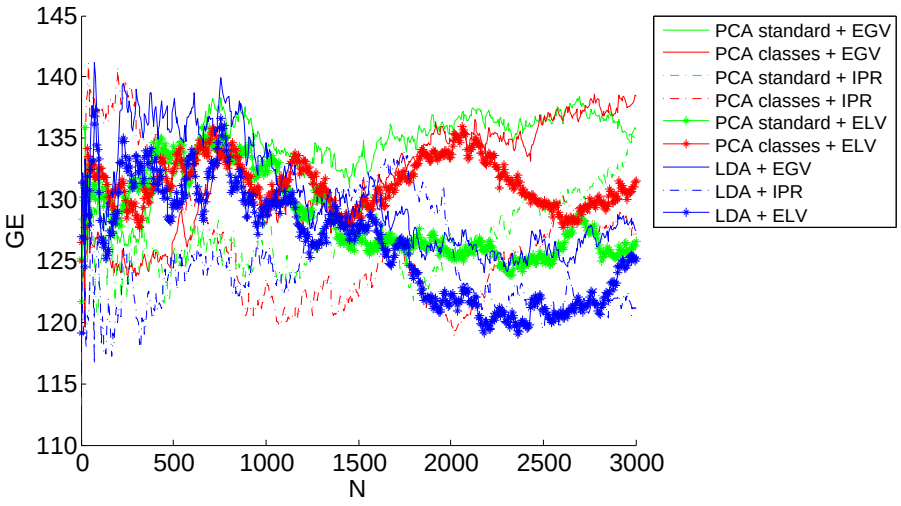


FIGURE 5.7: Degradation of linear-reduction-based template attacks in presence of misalignment.

Chapter 6

Kernel Dimensionality Reduction

6.1 Motivation

6.1.1 Higher-Order Attacks

Higher-Order Version of Projection Pursuits

6.2 Kernel Function and Kernel Trick

6.2.1 Local Kernel Functions as Similarity Metrics

6.3 Kernel Discriminant Analysis

6.4 Experiments over Atmega328P

6.4.1 The Regularization Problem

6.4.2 The Multi-Class Trade-Off

6.4.3 Multi-Class vs 2-class Approach

6.4.4 Asymmetric Preprocessing/Attack Approach

Comparison with Projection Pursuits

6.5 Drawbacks of Kernel Methods

Misalignment Effects

Memory Complexity and Actual Number of Parameters

Two-Phases Approach: Preprocessing-Templates

Chapter 7

Convolutional Neural Networks against Jitter-Based Countermeasures

ι

Chapter 8

Siamese Neural Networks for Collision Attacks

Chapter 9

Conclusions and Perspectives

9.1 Summary

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

Appendix A

Scenario 3 and 4 of CARDIS '15 paper

A.1 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. **??**, *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. 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. 5.6(f) it can be remarked that while the class-oriented PCA + ELV line keeps constant on the value 0 of guessing entropy, the class-oriented PCA + IPR is sometimes higher than 0.

Is the table with results overview interesting?

A.2 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., *we impose that the matrix A defining the extractor (see* **??***) has $C = 3$ rows (storing the 3 chosen components, transposed) and exactly $\sharp\text{PoI}$ non-zero columns. Looking at Fig.* **??** *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.*

Artificially Simulate Jitter

Bibliography

[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. 35, 36).

[Bat+11] Lejla Batina et al. “Mutual information analysis: a comprehensive study”. In: *Journal of Cryptology* 24.2 (2011), pp. 269–291 (cit. on p. 20).

[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. 19).

[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. 19).

[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. 20).

[BHK97] Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. 1997 (cit. on p. 36).

[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. 35, 36, 40).

[Bis06] Christopher M.. Bishop. *Pattern recognition and machine learning*. Springer, 2006 (cit. on pp. 20, 21, 27).

[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 p. 35).

[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. 35).

[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 p. 36).

[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. 35).

[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. 22, 35, 36, 39).

[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. 20, 21).

[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 p. 35).

[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. 35).

[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. 35).

[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).

[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).

[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).

[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).

[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 p. 36).

[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 p. 18).

[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. 35).

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. "Differential power analysis". In: *Annual International Cryptology Conference*. Springer. 1999, pp. 388–397 (cit. on p. 7).

[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 p. 7).

[Lib13] Joint Interpretation Library. *Application of Attack Potential to SmartSmart, Version 2.9*. Tech. rep. Common Criteria, 2013 (cit. on p. 13).

[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. 20).

[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. 36, 40, 41).

[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 p. 18).

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

[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. 23).

[Par] TELECOM ParisTech. "DPA Contest 4". In: (). http://www.DPAcontest.org/v4/. URL: http://www.DPAcontest.org/v4/ (cit. on p. 40).

[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. 21).

[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).

[RS09] Mathieu Renauld and François-Xavier Standaert. "Algebraic Side-Channel Attacks." In: *Inscrypt* 6151 (2009), pp. 393–410 (cit. on p. 22).

[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. 22).

[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 p. 35).

[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. 36, 40).

[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. 23).

[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 p. 36).