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

May 24, 2018

Contents

List of Figures

List of Tables

List of Abbreviations

SCA   Side Channel Attack
AES   Advanced Encryption Standard
NIST   National Institute of Standards and Technology
ANSSI   Agence National de la Sécurité des Systèmes d' Information
ITSEF   Information Technology Security Evaluation Facility
CESTI   Centre d'Evaluation de la Sécurité des Téchnologies de l'Information
EAL   Evaluation Assurance Levels
TOE   Target Of Evaluation
SFR   Security Functional Requirements
SAR   Security Assurance Requirements
CC   Common Criteria
ETR   Evaluation Technical Rapport
EGV   Explained Global **Variance**
ELV   Explained Local **Variance**
PCA   Principal Components Analysis
LDA   Linear Discriminant Analysis
KDA   Kernel Fisher Discriminant Analysis
SSS   Small Sample Size problem
NN   Neural Network
CNN   **Convolutional** Neural Network
DA   Data Augmentation

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. The readable message is referred to as *plaintext*, while the unintelligible output of the encryption is referred to as *ciphertext*. In general, cryptography aims to construct protocols to secure communication, while cryptanalysis studies the resistance of cryptographic techniques, developing *attacks* to break the cryptosystems' security claims. These two complementary domains evolve in 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 is seen as a branch of different disciplines, such as applied mathematics, computer science, electrical engineering, and communication science. Modern cryptosystems exploit algorithms based on mathematical tools and are implemented as computer programs, or electronic circuits. Their goal is to provide security functionalities for communications that use *insecure channels*, for example the internet. In particular, modern cryptosystems are designed in order to ensure at least one of the four following information security properties:

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

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

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

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

Two branches of cryptography may be distinguished: the *symmetric cryptography* and the *asymmetric cryptography*. The first one historically appeared before and is based on the hypothesis that the two communicating entities share a common secret, or 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 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 often cited in this thesis is provided; 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 called the *state*, denoted s. Such a state has 4 rows and 4 columns, thus contains 16 bytes. The byte lying at the i-th row, j-th column of s will be denoted by $s_{i,j}$ for $i, j \in \{0, 1, 2, 3\}$. The 16 input bytes and the 16 output bytes are indexed column-wise as shown in Fig. 1.1. Each element $s_{i,j}$ of a state is mathematically seen as an element of the *Rjindael finite field*, defined as $GF(2^8) = \mathbb{Z}/2\mathbb{Z}[X]/P(X)$ where $P(X) = X^8 + X^4 + X^3 + X + 1$. Five functions are performed during the AES, named KeySchedule, AddRoundKey, SubBytes, ShiftRows and MixColumns. At high level the AES algorithm is described hereafter:

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

Round 0:

AddRoundKey

Rounds 1 to penultimate:

SubBytes
ShiftRows
MixColumns
AddRoundKey

---

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

FIGURE 1.1: State array input and output. Source: [NIS].
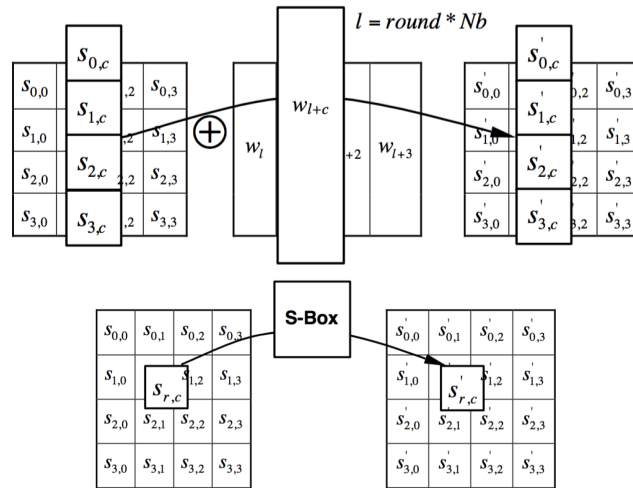
FIGURE 1.2: AddRoundKey (top) and SubBytes (bottom) operate over the State byte by byte, independently. Source: [NIS].

Last Round:

> SubBytes
>
> ShiftRows
>
> AddRoundKey

A description of the five functions is provided hereafter.

AddRoundKey

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

SubBytes

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

- the inversion in $GF(2^8)$ where the element $\{00\}$ is mapped to itself

- the affine transformation which maps each byte b_i to:

$$b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i , \qquad (1.1)$$
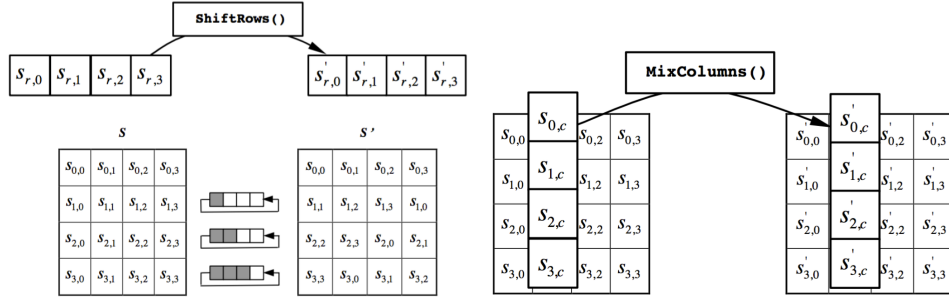
FIGURE 1.3: ShiftRows operates over the State rows. MixColumns operates over the State columns. Source: [NIS].

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

ShiftRows

The bytes in the last second, third and fourth rows of the State are cyclically shifted over 1, 2, and 3 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 $a(X) = \{03\}X^3 + \{01\}X^2 + \{01\}X + \{02\}$.

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] and Sbox is the substitution table used for the SubBytes transformation.

1.1.2 Description of RSA

The RSA cryptosystem, proposed in 1978 by Rivest, Shamir and Adleman [RSA78], 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 [DH76]). RSA is one of the most deployed, analysed, challenged, and discussed cryptosystems of all times, the seminal paper [RSA78] counting today $18,950$ citations.[3]

---

[2]where $\{02\} = (00000010)_2$ is represented by the polynomial x
[3]From Google Scholar, visited on January 2018.

Hereafter a simplified description of the RSA algorithm is provided.

Let $N = p \times q$ be the product of two large primes of about the same size. Then we can compute two values e, d such that e and $\varphi(N)$ are co-prime and $d = e^{-1}(\text{mod } \varphi(N))$, where $\varphi(N) = (p-1) \times (q-1)$ is the Euler's totient function of N, corresponding to the size of the multiplicative field $\mathbb{Z}_N^\star$. The pair (N, e) is the public key, and d is the private one. Let $M \in \mathbb{Z}_N^\star$ be a message. M can be encrypted by computing the value $C = M^e(\text{mod } N)$. The ciphertext C can then be decrypted by computing $C^d(\text{mod } N)$. The fact that the two transformations are mutually inverse is a consequence of the Euler's totient theorem. As e denotes the public exponent, and d the private one, the transformation $C = M^e(\text{mod } N)$ is called encryption as only the owner of the private key can retrieve the initial message. On the other hand $S = M^d(\text{mod } N)$ is called signature as it only needs public parameters to retrieve the initial message and thus anyone can verify that the owner of the private key has *signed M*. The security of such a cryptosystem is based on the big integers factorization problem.

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

Today, about 40 years after its invention, they still have a huge diffusion, both in terms of applicative domains and in terms of quantity of exemplars. Indeed, they serve as credit or ATM cards, healthy cards, ID cards, public transport payment cards, fuel cards, identification and access badges, authorization cards for pay television, etc. Slightly changing the card support, we find other applications of the same kind of integrated circuits, for example the mobile phone SIMs (*Subscriber Identity Module*) and the electronic passports. In terms of quantity, 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.

TABLE 1.1: Classification of Attacks

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

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 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-sized architecture, such as smart cards, were vulnerable to this kind of Side-Channel Attacks, the last cited recent work about acoustic emanations, together with other works exploiting electromagnetic fluctuations pointed out that much faster and bigger devices, *i.e.* laptops and desktop computers, are vulnerable as well [Gen+15; GPT15; Gen+16].

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 physical attacks that can be performed to break cryptographic devices security claims. A classification for such attacks is briefly outlined in Tab. 1.1. 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;

FIGURE 1.4: The actors of French Certification Scheme

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 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).

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 actors

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

- **The Developer**, who conceives a product and wishes to sell it as a certified secured product. He sends a request for evaluation to the certification body and, once the request is accepted, he contacts an evaluation laboratory;

- **The ITSEF** is the *IT Security Evaluation Facility*; in France it is named *Centre d'Evaluation de la Sécurité 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 Sécurité des Systèmes d'Information* (ANSSI) in France, or the *Bundesamt für Sicherheit in der Informationstechnik* (BSI) in Germany, for example. It ensures the quality of the evaluation and delivers a certificate to the developer;

- **The end user**, who buys the product and follows its security guidelines.

The Target of Evaluation and the security objectives

To start the certification process, the developer compiles a document called *Security Target* (ST). Such a document begins specifying the (part of the) device subjected to evaluation, the so-called *Target of Evaluation* (TOE), then lists its *Security Functional Requirements* (SFR), choosing 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*, associated with the required SFRs; for example *smart card* or *TEE* designate some precise Protection Profiles.

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

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

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 needs to respect the EMVco norms, being EMVco a consortium of six companies (Visa, MasterCard, JCB, American Express, China UnionPay, and Discover) that manages private certification schemes for banking cards, payment terminal and automated teller machines.

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 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. This is a list of the possibly dangerous attack paths, basing on a code analysis, and on the state-of-the-art attacks list in general provided by working groups dedicated to the 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 kinds of attacks. The guidelines for the cotation table are given by the *Common Methodology for Information Technology Security Evaluation* (CEM).

In the case of smart cards the evaluation systematically includes the AVA_VAN5 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* (JHAS), a subgroup of the working committee *Senior Officials Group Information Systems Security* (SOG-IS) which coordinates the standardization of CC. Moreover, the JHAS produces the *Application of Attack Potential to Smartcards* [Lib13] of the JIL, which is an interpretation of the CEM in the special case of smart cards. The cotation table factors specified by the JHAS are detailed in Table 1.5. The evaluation is divided in two parts, an *identification* part, that reflects the difficulty in finding the attack path, and an *exploitation* part, that reflects the difficulty in actually performing the attack. The total score of an attack is the sum of scores assigned to each factor. To obtain the AVA_VAN5 grade every attack tested by the evaluators must have been rated at least 32.

The Evaluation Technical Report

The evaluation ends with the redaction by the evaluators of an *Evaluation Technical Report* (ETR), which is transmitted to the certification body. 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 vulnerabilities are kept as *residual vulnerabilities*. The ETR is strictly reviewed annually

---

[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

by the evaluators in charge of the surveillance of the certificate. For the penetration testing, the evaluators are in particular asked each year to verify that the cotation of the attacks presented in the ETR did not drop.

1.3 This thesis objectives and contributions

Among the factors observable in the cotation table 1.5 we find *open samples*, 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 loaded specific software. The evaluator generally exploits this possibility to ease the task of characterization of the physical behaviour of the device.

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

1.3.1 Foreword of this Thesis: Research of Points of Interest

To practice a Side-Channel Attack, the monitoring of unintentional channels leaking from the attacked device is usually performed through an oscilloscope that samples continuous analog signals and turns them into discrete digitalized sequences. Such sequences are often referred to as *traces*. To allow a deep inspection of the device, the sampling rate of the oscilloscope needs to be high, leading very often to a high dimensionality of such traces. Nevertheless, it is expected that only a limited number of time samples are relevant for a SCA: those that are statistically dependent on the sensitive variable that is exploited to run the attack. Such time samples are called *Points of Interest* (PoIs). In literature a few of different statistics was proposed and exploited to choose such PoIs in a preliminary attack phase, in order to reduce both time and memory complexity of the attacks. A brief overview of such statistics is proposed in Sec. 2.1. The foreword of this thesis was to propose new methods to research and characterise such PoIs, in order to ameliorate and possibly optimise the preliminary attack phase consisting in their selection.

1.3.2 Dimensionality Reduction Approach

Beyond the use of point-wise statistics to identify the PoIs, an axe of research was launched in SCA context, importing from the Machine Learning domain more general techniques for dimensionality reduction of data. At the moment this thesis researches begun, linear methods were drawing a raising attention, consisting in techniques to conveniently exploit linear combinations of many time samples. The first contributions we proposed belong to this axe of research: in Chapter 4 we describe the two mainly deployed techniques, the Principal Components Analysis and the

Linear Discriminant Analysis, and tackle some open issues about their application to SCA context. The solutions proposed in the thesis have been presented at CARDIS 2015 [CDP16a] and published in the proceedings of such an international conference.

Nowadays every device needing to obtain an AVA_VAN5 grade is equipped of specific countermeasures against SCAs. A brief overview of some classic and public principles providing efficient countermeasure is provided in Sec. 2.2. Among them, the *masking*, or *sharing*, countermeasure besides being the most popular and considered the most efficient, is the most likely to require an adaptation of the attack strategy in order to be defeated. Indeed, when an effective masking scheme is implemented, each sensitive variable of the original computation is divided probabilistically into shares such that any proper subset of shares is statistically independent of the sensitive variable itself. Computation of cryptographic primitives is done accessing only the random shares, with intermediate steps computing only the shares of the result. This forces the attacker to work with the joint distributions of the signal at the time samples where the shares are being accessed. In other words, point-wise statistics to retrieve PoIs are completely inefficient in presence of a masking countermeasure, since each time sample is by itself statistically independent from any sensitive variable. Moreover, interesting joint distributions have to be studied at their higher-order statistical moments to retrieve sensitive-data dependencies, implying that any linear method to combine time samples is inefficient as well. To sum up, the issue of selecting PoIs or applying dimensionality reduction to side-channel traces protected by masking presents challenging difficulties. Such a hardness is mitigated when the attacker is able to perform a profiling phase during which he has the knowledge of the random values assigned to the shares during execution. In this thesis we tackle the issue when such knowledge is not enabled, and we propose in Chapter 5, on the base of a work presented at CARDIS 2016 [CDP16b], to deploy Kernel Fisher Discriminant Analysis (KDA) as a solution. This is an extension of the LDA dimensionality reduction technique, allowing applying some strategy to combine non-linearly time samples.

1.3.3   Neural Networks Approach

Kernel techniques like the KDA are as well inherited from Machine Learning domain and consist in strategies that allow to build interesting extensions of many algorithms. One of their characteristic, that turns to be a drawback to apply them in SCA context is that are memory-based: the entire set of profiling traces, *i.e.* those acquired by observing the open samples, has to be stored and accessed in the attack phase. In this sense they are highly memory-consuming, and quite slow to apply: they do not scale well in presence of huge profiling trace sets as those that are often necessary to perform profiling SCAs. In contrast to them, models provided by Neural Networks (NNs) are Machine Learning solutions that are known to be easily scalable to huge datasets and not memory-based. We decided to explore such an approach and pointed out that it not only provided solutions to tackle such a computational performance drawback. Neural Networks are models that permits to switch from a Machine Learning to a Deep Learning paradigm: they allow to deal with profiling attacks in an integrated way that avoids separate preprocessing phases as the dimensionality reduction or the selection of PoIs from the proper characterisation part. They are non-linear models, implying that they are able to deal with side-channel traces protected by masking countermeasure. Moreover, some special structures of Neural Networks, the so-called Convolutional Neural Networks (CNNs), originally

conceived for image recognition application, fit well to handle other kinds of classic countermeasures: those improving trace desynchronization, or misalignment (see Sec. 2.2). Classically, misalignment had to be treated by other special preprocessing phases, that Deep Learning paradigm allows to integrate in a single learning process as well as dimensionality reduction. In Chapter 6, on the base of the publication presented at CHES 2017, we discuss about the advantages of exploiting such CNNs in SCA context. As we will see, in this case we equipped our study with another technique that is classical in Machine Learning domain: the Data Augmentation (DA).

1.3.4   A bridge between Side-Channel Attacks and Machine Learning

As a general observation about the track we followed during this thesis, we started from the problem of identifying the PoIs in a signal, that is classically tackled by means of pure statistical tools, as hypothesis tests, then enlarge both the objectives and methodology classes. We addressed more general and complex issues as the dimensionality reduction, first in absence then in presence of masking countermeasures, than the still more large *classification* task in SCA context. To do so we explored more and more complex models accepting passing from a formal statistical asset to the Machine Learning one, which carries with him some intrinsic non-optimality, formalised by the so-called *No Free Lunch theorem*, briefly stated in Sec. 3.1.6. In last decades Machine Learning domain is progressing very fast, and drawing attention in many application contexts. In last years a transfer from Machine Learning to the application domain of SCA started, and our researches make parts of such a flow. We believe that many kinds of side-channel scenarios, and especially profiling contexts, may be turned into a Machine Learning language and many researches already carried out for other applications should be exploited to understand if they represent or not a danger in embedded security domain, leading to powerful SCAs. In this sense and by way of example, we propose in Chapter 7 the analogy between a known class of SCAs, the so-called *collision attacks*, and the Machine Learning task known as *verification*, and tackle such a scenario with a special kind of Neural Network structure, the so-called *siamese network*. The promising preliminary results we obtained demonstrates the interest in exploring attentively the Machine Learning researches in order to ameliorate SCAs and, by consequence, adapt secured devices countermeasures, and enhance reliability of cryptographic hardware.

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

Chapter 2

Introduction to Side-Channel Attacks

2.1 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*, making the complexity of the attack drop significantly.

Remark. The core of an RSA implementation is the computation of a modular exponentiation with base, modulo and exponent being some big numbers (*e.g.* 1024 bits). A myriad of different algorithms have been proposed to implement in hardware the RSA, which differ from each other for many factors: the way of representing data (*e.g.* Montgomery representation, binary representations with signed bits, Chinese Reminder Theorem representation . . .), the way of scanning the exponent (*e.g.* scan the exponent bit by bit, or window by window, from right to left or left to right, priorly decomposing the exponent or not, . . .), the way of performing in hardware the multiplication (*e.g.* schoolbook, Montgomery, Barrett, . . .). Each choice impacts the size of the needed hardware and the time of the computation. A quite exhaustive state-of-the-art of such implementations is provided in [Koc94].

Sensitive Variable

In order to make inference on the 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 most classical choice to perform a so-called *differential* or *advanced* SCA (see below);

- any function of a cryptographic variable (ex: $\mathrm{HW}(f(K, E))$), where 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 intermediate operations in a Montgomery ladder implementation of RSA)

In this thesis we will try as much as possible to abstract from the form of the sensitive variable, thinking of any entity Z that assumes values in a finite set $\mathcal{Z}$ and whose value 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 a sensitive variable of the implemented algorithm leak during its execution through some observable *side* channels. Such leakages are collected in the form of signal traces. Depending on the observed channel (*e.g. power consumption, electromagnetic irradiation, time, . . .*), different properties might influence the form of the leakage, and should be taken into account for the construction of a leakage model.

If we allow a Side-Channel attacker to make use of a probing station to directly access the circuit wires and monitor the exact values of some intermediate values, this attacker will observe leakages following the so-called *probing model*. To define this model no further hypothesis 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. 2.2.2).

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, which consume depending on how many bits set to 1 or 0 they store or transport (Hamming weight) or how many of them switch their value for 0 to 1 or vice-versa during computations (Hamming distance).

When an attacker has chosen its sensitive target Z and deals with concrete acquisitions, he does not need a complete power model, but only a way to 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* [Cha+99]. 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 $B \approx \mathcal{N}(0, \sigma^2)$.

Remark. in this model the leakage is a real value X. In general leakages are acquired as traces, represented by column vectors $\vec{X} \in \mathbb{R}^D$ and model 2.1 refers to a single well-chosen *point of interest* (see later for a definition).

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 (see for instance [Bat+16]). Algebraic SCAs (see below) are horizontal attacks as well, in which inferences are not done over a single sensitive variable, but potentially over every cryptographic variable, followed by a deep analysis of 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 $(\mathbf{x}_i, z_i)_{i=1,...,N_p}$ for which the correct association trace/sensitive variable is known. The second phase of a profiling attack is the proper *attack phase*, during which the attacker observes a new set of acquisitions, under unknown secret key, and 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 studied under the name of *supervised machine learning*. In supervised machine learning couples $(\mathbf{x}_i, z_i)_{i=1,...,N_p}$ 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 $z \in \mathcal{Z}$. In such a context N_z will denote the number of profiling traces belonging to the class z.

Simple vs Advanced SCAs

A simple attack is an attack that only needs 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.

Remark. Attacks for which many observations are acquired with fixed entry parameters and by consequence in which the observed leakage always corresponds to a not variable Z are still considered simple attacks. The attacker may exploit 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

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

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

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

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):

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

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. 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 the 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 Gaussian 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.

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 4.

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

---

[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].

one provided by non-interesting points. The goal of researches in this context is to study and develop techniques to characterize PoIs and to apply convenient dimensionality reduction techniques, that allow reducing the size of the acquisitions while keeping the exploitable information held by data high enough to allow an SCA to succeed. Representing the side channel traces as column vectors $\mathbf{x}$ in $\mathbb{R}^D$, the compressing phase might be seen as the application of a function $\epsilon \colon \mathbb{R}^D \to \mathbb{R}^C$, with $C < D$, called *extractor* throughout this thesis. The first extractors proposed in SCA literature where actually some selectors of time samples, *i.e.* functions that operate a simple subsampling of the traces on the base of the computation of some sample-wise statistics $\tau(t)$, whose aim is to quantify a sort of signal strength. Several proposals exist for such a signal-strength estimate, among them the most deployed are the Difference of Means (DOM) [CRR03], or the analogous but better specified Sum of Differences (SOD) [RO05], the Sum of Squared Differences (SOSD) [GLRP06], the Signal-to-Noise Ratio (SNR) [MOP08; LPR13] and Sum of Squared t-differences SOST, corresponding to the t-test [GLRP06]. All these statistics are similar, and exploit the sample mean per class of the traces, given by

$$\bar{\mathbf{x}}_z = \frac{1}{N_z} \sum_{i \colon z_i = z} \mathbf{x}_i \,. \tag{2.7}$$

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

$$\hat{\boldsymbol{\sigma}}_z^2 = \frac{1}{N_z - 1} \sum_{i \colon z_i = z} (\mathbf{x}_i - \bar{\mathbf{x}}_z)^2 \,. \tag{2.8}$$

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

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

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 such seminal papers block ciphers implementations are attacked, and the authors added to plaintext and ciphertext knowledge, classically assumed in algebraic cryptanalysis, the access to the (exact) Hamming weights of several intermediate computations, obtained by side-channel observation. Once recovered as many partial information as possible, it is expressed in the form of an equation system, then converted to a set of clauses treatable by a SAT solver. In opposition to classic SCAs, ASCA approach does not exploit a divide-and-conquer strategy: the

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

Name of the estimate	Definition
SOD	$\tau(t) = \sum_{\substack{z_1, z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\overline{\mathbf{x}}_{z_1}(t) - \overline{\mathbf{x}}_{z_2}(t))$
SOSD	$\tau(t) = \sum_{\substack{z_1, z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\overline{\mathbf{x}}_{z_1}(t) - \overline{\mathbf{x}}_{z_2}(t))^2$
SOST (version [GLRP06])	$\tau(t) = \dfrac{\sum_{\substack{z_1, z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\overline{\mathbf{x}}_{z_1}(t) - \overline{\mathbf{x}}_{z_2}(t))^2}{\dfrac{\hat{\boldsymbol{\sigma}}_{z_1}^2}{N_{z_1}} + \dfrac{\hat{\boldsymbol{\sigma}}_{z_2}^2}{N_{z_2}}}$
SOST (version [BDP10])	$\tau(t) = \dfrac{\sum_{\substack{z_1, z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\overline{\mathbf{x}}_{z_1}(t) - \overline{\mathbf{x}}_{z_2}(t))^2}{\hat{\boldsymbol{\sigma}}_{z_1}^2 + \hat{\boldsymbol{\sigma}}_{z_2}^2}$
SNR	$\tau(t) = \dfrac{\hat{\mathrm{Var}}_z(\overline{\mathbf{x}}_z(t))}{\hat{\mathbb{E}}_z[\hat{\boldsymbol{\sigma}}_z^2(t)]}$ $\qquad$ (2.9)

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 [OWW13] 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.

Efficiency of the SCAs

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

$$\mathrm{Rank}(k^\star) = i \text{ such that } \mathbf{g}[i] = k^\star. \tag{2.10}$$

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

$$\mathrm{SR}_o = \mathrm{Prob}[\mathrm{Rank}(k^\star) \leq o] . \tag{2.11}$$

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

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

$$\mathrm{GE} = \mathbb{E}[\mathrm{Rank}(k^\star)] . \tag{2.12}$$

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

2.2 Main Side-Channel Countermeasures

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

---

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

2.2.1 Hiding

The main characteristic of a hiding countermeasure is that is does not change the intermediate data values that are processed in the cryptographic algorithm, but it only attempts in hiding its processing. Hiding is typically, but not only,[3] achieved in by randomising the power consumption. A random power consumption can be obtained by randomly changing the time at which the targeted sensitive variable is processed. In this way the attacker acquires side-channel traces that are desynchronised or misaligned with respect to their interesting part. This temporal misalignment reduces the effectiveness of an attacker's statistical analysis. Possible ways for randomising the power consumption are the random insertion of dummy instructions [CK09; CK10] and the shuffling of the operations [VC+12], at a software level, or the randomization of the instruction stream by means of non deterministic processors [IPS02; MMS01], or the enhancement of a jittering effect over the clock via an asynchronous logic style at a hardware level [Moo+02; Moo+03]. Such methods may also be combined.

Applying realigning preprocessing techniques, such as integration, pattern matching or approaches based on frequency analysis, is the most common approach an attacker usually chooses to face up temporal misalignment. Defeating differently misalignment countermeasures is one of the main motivations that lead us to investigate Convolutional Neural Networks, as we will discuss in Chapter 6.

2.2.2 Masking

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

$$Z = M_1 \star \cdots \star M_d \tag{2.13}$$

holds for a group operation $\star$ (*e.g.* the exclusive or for the most popular Boolean masking already proposed in the seminal papers [Cha+99; GP99]). The soundness of the masking countermeasure is implied by the fact that, in the noisy leakage model, the complexity of recovering information by SCA on a bit shared into several pieces grows exponentially with the number d of shares.[4] This fact was enlighten by Chari *et al.* in 1999 [Cha+99], then complemented by Prouff and Rivain in 2013 [PR13]. As a consequence of such an exponential complexity behaviour, the number d of shares

---

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

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

plays the role of a security parameter for a masking scheme and the method is usually referred to as $(d-1)$th-order masking (since it involves $d-1$ random values, called *masks* and one value determined by the sensitive variable and the relation (2.13), which is sometimes referred to as *masked variable*). The shares are manipulated by distant parts of the circuit (especially if the countermeasure is implemented at a hardware level) or at different times (especially for software implementations of the countermeasure). In this way an attacker, who is obliged to retrieve information coming from a sufficient number of shares to obtain some Z-dependent information, has to acquire many portions of signal to combine.

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

Chapter 3

Introduction to Machine Learning

3.1 Basic Concepts of Machine Learning

Machine Learning (ML) is a field of computer science that groups a variety of methods whose aim is giving computers the ability of *learning* without being explicitly programmed. The more cited definition of *learning* has been provided by Mitchell in 1997 [TM97]: " A computer program is said to learn from experience E with respect to some task T and performance measure P, if its performance on T, as measured by P, improves with experience E."
Machine Learning groups a variety of methods essentially coming from applied statistics, and 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 often said to "learn from data", in the sense that it is able to improve a computer program's performance at some task via a data observation experience.

3.1.1 The Task, the Performance and the Experience

The task 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* $\mathbf{x}[i], \quad i = 1, \ldots D$. In SCA context an example might be 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 $f : \mathbb{R}^D \to \mathcal{Z}$. We remark that this task is similar to the regression one, except for the form of the output, since in general $\mathcal{Z}$ is a discrete finite set. A slightly variant solution to the classification task consists in constructing a function $f : \mathbb{R}^D \to \{0,1\}^{|\mathcal{Z}|}$, 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 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

signatures have been produced by the same person. The learning algorithm is thus asked to construct a function $f \colon \mathbb{R}^D \times \mathbb{R}^D \to \{0, 1\}$. A variant of such a task consists in finding 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 form 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 instructor 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*. For example, the Principal Component Analysis, that will be discussed in Chapter 4 in the context of SCA, is a dimensionality reduction algorithm that might be seen as an unsupervised algorithm that learns a representation of data. We will see that 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, such as the underfitting and the overfitting (see 3.1.4). Let us introduce a linear regression model to tackle the regression task: we want to construct a linear

function $f\colon \mathbb{R}^D \to \mathbb{R}$, that takes an input $\mathbf{x}$ and outputs $\hat{y} = \mathbf{w}^\intercal \mathbf{x}$, 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. Let us 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 $\hat{y} = \mathbf{w}^\intercal \mathbf{x}$, 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. When a learning algorithm behaves as an optimization algorithm that minimizes a given function, such a function is called *cost function*, of *objective function*. The solution to such a optimization problem can be given in closed form, by means of the pseudo-inverse matrix of $\mathbf{M}_{\text{train}}$:

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

3.1.3 Example of Linear Model for Classification

In this thesis we point out a strict relationship between the profiling SCAs and the classification task in Machine Learning context. For this reason we introduce here a very brief overview of how classically such a task is tackled, by means of linear models.

Classifying means assigning 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 6, 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

---

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

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 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

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

Under this assumptions Eq. (3.4) rewrites as:

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

where we set

$$\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2)$$

$$w_0 = \log\frac{p(z^1)}{p(z^2)} - \frac{1}{2}\mu_1^\mathsf{T}\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^\mathsf{T}\Sigma^{-1}\mu_2.$$

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 $\mathbf{w}^\mathsf{T}\mathbf{x}$ 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 4, to which we will refer to LDA dimensionality reduction by a widely accepted 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):

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

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 6 when considering neural networks as models.

3.1.4 Underfitting, Overfitting, Capacity, and Regularization

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

---

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

generalization gap, is too large.

Capacity   The property of a model that controls its underfitting or overfitting be-
haviour is the *capacity*. Roughly speaking the capacity of a model quantifies the
complexity of the functions it can represent: a model with higher capacity can be
parametrized is such a way to represent a higher complex function. For example a
linear regression model is able to represent all linear functions. To raise its capac-
ity quadratic, cubic or general polynomial terms might be included, to allow the
model represent respectively quadratic, cubic or polynomial functions as well. An-
other common way to enlarge the capacity of a linear regression model $y = \mathbf{w}^\mathsf{T}\mathbf{x}$,
consists in choosing some *basis functions* $\varphi_1, \varphi_2, \ldots, \varphi_B$ and replace $\mathbf{x}$ with the values
$\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_B(\mathbf{x})$. The form of the basis functions will determine the capacity
of the model. Basis function regression includes the linear and the polynomial case.

   The polynomial regression provides a striking example to understand the under-
fitting and overfitting phenomena. Consider a problem in which examples (x, y) lies
in $\mathbb{R} \times \mathbb{R}$ and the true underlying function is quadratic, perturbed by a small noise.
Let the training set contain 4 data points. Figure 3.1 shows the results of a linear,
quadratic and cubic regression in such a case. We can observe that the linear pre-
dictor is underfitting, since its training error is quite high, while the cubic predictor
is overfitting: it perfectly fits the 4 training points (it is the Lagrange polynomial in-
terpolating such 4 points) but shows a huge error in predicting new examples. The
quadratic regression is obviously in this case the model exhibiting the optimal ca-
pacity to solve such a problem. A very rough way to have an intuition about the
capacity of a model is counting the number of its parameters: the capacity in general
grows with the number of parameters. In our polynomial regression example the
cubic model has as many parameters as the number of training points.
Some formal ways to quantify the capacity of a model have been provided in ML lit-
erature. The most well-known is the *Vapnik-Chervonenkis dimension*: it measures the
capacity of a classifier as the cardinality of the largest set of points the model can clas-
sify without errors, for any possible assignment of labels. In practice, quantifying
the capacity of a model, especially for complex models as those considered in deep
learning, is very hard and discouraged. On the other hand, these kinds of quantifi-
cations has enabled statistical learning theory to formalize and prove some impor-
tant intuitions, for example the fact that the generalization gap is upper-bounded
by a quantity that grows with the model capacity and that shrinks as the number of
training examples increases. In Fig. 3.1(d) we observe how the cubic model used for
regression on quadratic distributed data ameliorate its performances and reduces
the generalization gap despite its excessive capacity, when trained with more ex-
amples. This observation basically justifies on one hand the deep learning attitude
consisting in consider very complex models, having confidence in the big size of
the typically considered training sets. On the other hand it justifies the interest of
Data Augmentation (DA) techniques to respond to an eventual lack of data. Some
DA techniques wll be proposed in Chapter 6 for the SCA context.

Regularization   In a real-case problem, the optimal capacity necessary to learn from
given data is unknown. In such a case, trying to fit data with a too low capacity
model assures the defeat, thus it is always more interesting to oversize the capacity
of the learning model. Choosing an oversized model, we risk to incur in overfitting.

MSE_train=44.228280, MSE_test=330.984916
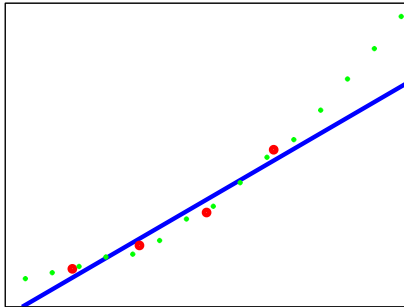
MSE_train=2.243097, MSE_test=61.891672
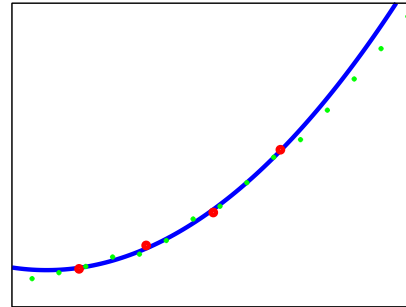
(a) Linear

(b) Quadratic

MSE_train=0, MSE_test=970.081580
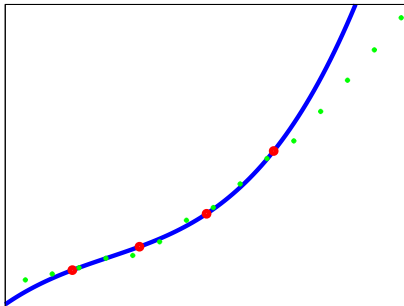
MSE_train=3.040333, MSE_test=58.377719
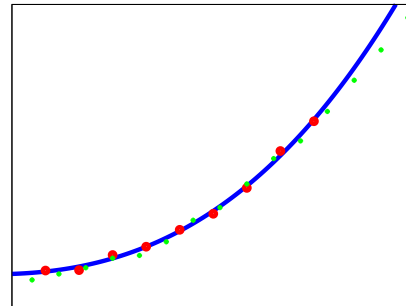
(c) Cubic

(d) Cubic, more training data

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

The so-called *regularization* techniques respond to such a risk: they consist in general in adding constraints to the learning algorithm in order to guide it in choosing a model among a wide set of eventually fitting models. Going back to the polynomial regression example, one can try to fit data with a cubic polynomial (thus oversizing the model capacity) and induce the optimizer algorithm to choose the smallest-degree polynomial fitting data be chosen via a regularization. This can be obtained adding a penalty that depends on the polynomial degree to the cost function. Applying regularization may make the algorithm be less accurate in learning training data, but more likely to correctly operate on new examples.

3.1.5   Hyper-Parameters and Validation

The *hyper-parameters* of a model are all parameters that are priorly set and that are not learned by the learning algorithm. They define the general form of the model. In the polynomial regression example the model had a single hyper-parameter: the degree of the polynomial. It is evident from the example that such a parameter is somehow forced not to be optimized by the means of the learning algorithm: trying to reduce the training MSE, the algorithm would choose a sufficient high degree to interpolate all training points. This would cause overfitting. In general among all parameters of a model, the hyper-parameters are chosen as those that can not be learned from data because it would cause overfitting, as in the example, or because they are too difficult to optimize.

A way to choose a setting for hyper-parameters consists in perform a *validation* phase. To do so the training set is split into two disjoint sets, one still called *training set* and the other one called *validation set*. We can say that as the training set is used to learn the parameters, the validation set is used to somehow learn the hyper-parameters. Indeed during or after the training over the training set, the validation set is used to estimate the test error, which quantifies the generalization ability of the model. In practice the performances of the (partially) trained model are evaluated over the validation set and hyper-parameters are updated accordingly, in order to raise the generalization ability of the model. Once the model has been validated, *i.e.* the hyper-parameters are definitely set, the real test error is evaluated over the test set. Usually the validation error is an underestimation of the test error, since hyper-parameters have been set to reduce it.

The validation process just described may strongly depend on the way the training set have been split to create the validation one. In order to avoid to validate a model in a strongly data-dependent way, a slightly different process is encouraged in machine learning community, named the *cross-validation*, which we describe in Appendix C.

3.1.6   No Free Lunch Theorem

A so-called *No Free Lunch Theorem* has been formulated for optimization and machine learning algorithms around 1997 [WM97]. It states that any learning algorithm, averaged over all possible data distributions, has the same performances over unseen test data. This means that there cannot exist a universal best machine learning algorithm: any of them performs in the same way, when performances are averaged over all possible tasks. Thus, making research over some kind of data, for example SCA traces, means trying to understand what kinds of machine learning

algorithms perform well over such particular kind of data and point out the eventual interesting hyper-parameters of machine learning models that are responsible of the main performance variations.

3.2 Overview of Machine Learning in Side-Channel Context

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

Machine Learning algorithms started to be investigated in in Side-Channel Attacks context in 2011 [Hos+11]. In this paper the authors formulated for the first time an attack in terms of classification problem and proposed the Support Vector Machine (SVM) [CV95; WW98] as technique to solve it. They also equipped the SVM with a kernel function to allow the it to succeed even in case data would not be linearly separable. Such an approach is similar to the one the we will describe in Chapter 5, to obtain Kernel Discriminant Analysis dimensionality reduction technique from the Linear Discriminant Analysis. Further works analysed the use of SVM in SCA context, proposing concrete attack scenarios [HZ12; BLR13]. The technique of Random Forest [Lio+14] drew attention of the SCA community as well. As the SVM, it has been used as a classifier and evaluated in different works [LBM15; Ler+15; LBM14]. As in recent years the privileged tools to tackle classification problem in Machine Learning area are the Neural Networks, whose multi-layer configuration has given name to the so-called *Deep Learning* domain, such tools have as well be analysed in SCA context. Networks in the form of Multi-Layer Perceptrons (MLP) have been proposed as classifiers for side-channel traces in a series of works [MHM13; MZ13; MMT15; MDM16], while Convolutional Neural Network was firstly introduced in [MPP16]. A part of this thesis contributions consists in the application of the convolutional paradigm as way to defeat misalignment countermeasures in side-channel attacks (see Chapter 6).

Part II

Contributions

Chapter 4

Linear Dimensionality Reduction

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

4.1 Introduction

Linear dimensionality reduction methods produce a low-dimensional linear mapping of the original high-dimensional data that preserves some feature of interest in the data. An abundance of methods has been developed throughout statistics, machine learning, and applied fields for over a century, and these methods have become indispensable tools for analysing high dimensional, noisy data, such as side-channel traces. Accordingly, linear dimensionality reduction can be used for visualizing or exploring structure in data, denoising or compressing data, extracting meaningful feature spaces, and more. A very complete survey about this great variety of linear dimensionality reduction technique has been published in 2015 by Cunningham and Ghahramani [CG15]. They proposed a generalized optimization framework for all linear dimensionality techniques, survey a dozen different techniques and mention some important 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 5.

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. 4.2) . In Sec. 4.3, 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 ELV selection tool has been tested in a successive experimental work [CK18]. Unfortunately, the authors of such a work could not observe an improvement (nor a worsening) using such a selector, because in their specific case it returned the selection of the first components by eigenvalues. The LDA technique will be described in Sec. 4.4, together with its previous uses in SCA literature, the description of the SSS
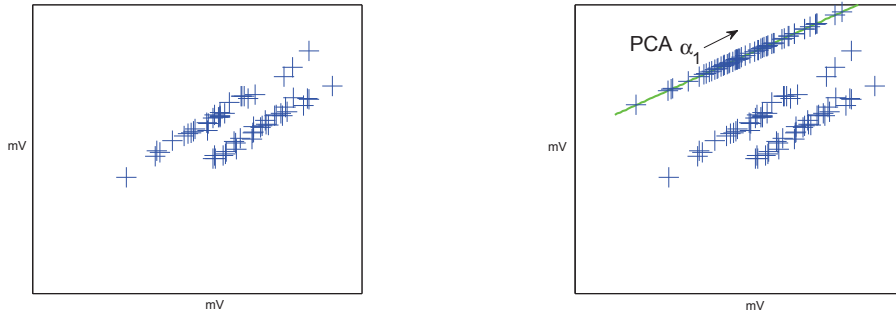
FIGURE 4.1: PCA: some 2-dimensional data projected into their 1-dimensional principal subspace.

problem and some solutions coming from the Pattern and Face Recognition communities [BHK97; Che+00; Hua+02; YY01]. Through some experiments depicted in Sec. 4.5 we will conclude about the effectiveness of the PCA-ELV solution. Finally, in Sec. 4.6 we will experimentally argue about the non robustness of all these techniques to data misalignment.

4.2 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 to project data onto a lower-dimensional linear space in such a way that the average projection cost, defined as the mean square distance between the data and their projections, is minimized. In the following it is shown how the PCA algorithm is deduced by the statistical definition. The reader interested by the equivalence between the two approaches can refer to [Bis06, Ch. 12]. An example of 2-dimensional data projected over their 1-dimensional principal subspace is depicted in Fig. 4.2.

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

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

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:

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

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

$$\frac{1}{N}\mathbf{y}_1\mathbf{y}_1^\mathsf{T} = \frac{1}{N}\boldsymbol{\alpha}_1\mathbf{M}\mathbf{M}^\mathsf{T}\boldsymbol{\alpha}_1^\mathsf{T} = \boldsymbol{\alpha}_1\mathbf{S}\boldsymbol{\alpha}_1^\mathsf{T} . \tag{4.3}$$

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:

$$\Lambda(\boldsymbol{\alpha}_1, \lambda) = \boldsymbol{\alpha}_1\mathbf{S}\boldsymbol{\alpha}_1^\mathsf{T} - \lambda(\boldsymbol{\alpha}_1\boldsymbol{\alpha}_1^\mathsf{T} - 1) \tag{4.4}$$

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

$$\frac{\partial\Lambda}{\partial\boldsymbol{\alpha}_1} = 2\mathbf{S}\boldsymbol{\alpha}_1^\mathsf{T} - 2\lambda\boldsymbol{\alpha}_1^\mathsf{T} . \tag{4.5}$$

Thus, stationary points of Λ verify

$$\mathbf{S}\boldsymbol{\alpha}_1^\mathsf{T} = \lambda\boldsymbol{\alpha}_1^\mathsf{T} , \tag{4.6}$$

which implies that $\boldsymbol{\alpha}_1^\mathsf{T}$ must be an eigenvector of $\mathbf{S}$, with λ its correspondent eigenvalue. Multiplying both sides of Eq. (B.2) by $\boldsymbol{\alpha}_1$ on the left, we remark that

$$\boldsymbol{\alpha}_1\mathbf{S}\boldsymbol{\alpha}_1^\mathsf{T} = \lambda\boldsymbol{\alpha}_1\boldsymbol{\alpha}_1^\mathsf{T} = \lambda, \tag{4.7}$$

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 $\mathbf{Y} = \left[\begin{smallmatrix}\boldsymbol{\alpha}_1\\\boldsymbol{\alpha}_2\end{smallmatrix}\right]\vec{X}$ 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

$$\Lambda = \boldsymbol{\alpha}_1\mathbf{S}\boldsymbol{\alpha}_1^\mathsf{T} + \boldsymbol{\alpha}_2\mathbf{S}\boldsymbol{\alpha}_2^\mathsf{T} - \lambda_1(\boldsymbol{\alpha}_1\boldsymbol{\alpha}_1^\mathsf{T} - 1) - \lambda_2(\boldsymbol{\alpha}_2\boldsymbol{\alpha}_2^\mathsf{T} - 1) . \tag{4.8}$$
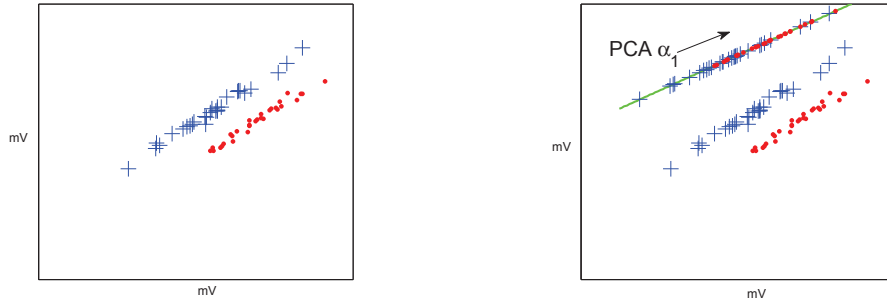
FIGURE 4.2: PCA: some 2-dimensional labelled data projected into
their 1-dimensional principal subspace.

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

$$\mathbf{S}\boldsymbol{\alpha}_1^\mathsf{T} = \lambda_1 \boldsymbol{\alpha}_1^\mathsf{T} \tag{4.9}$$
$$\mathbf{S}\boldsymbol{\alpha}_2^\mathsf{T} = \lambda_2 \boldsymbol{\alpha}_2^\mathsf{T}, \tag{4.10}$$

which means that $\boldsymbol{\alpha}_1^\mathsf{T}$ and $\boldsymbol{\alpha}_2^\mathsf{T}$ 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.

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

4.3 Application of PCA in SCAs

4.3.1 Original vs Class-Oriented PCA

The classical version of PCA method is unsupervised. 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 labelled set of traces $(\mathbf{x}_i, z_i)_{i=1..N_p}$.
In this context, and for the sake of distinguishing the target value assumed by the target Z in new executions, the idea of the *Class-Oriented* PCA is to consider *equivalent* all traces belonging to the same class. Modelling traces of a same class as traces showing the same characteristic form plus a random noise, the denoised characteristic form can be estimated by the sample means of the traces in the trace. Supposing that the profiling trace set is balanced, *i.e.* $N_p = |\mathcal{Z}| \times N$ for some N, the sample means are given by $\bar{\mathbf{x}}_z = \frac{1}{N} \sum_{i:\, z_i=z} \mathbf{x}_i$. In class-oriented PCA these sample means takes the place of rough data: we collect them as rows in the data matrix $\mathbf{M}$, shift the empirical mean of each component to zero, compute the covariance matrix $\mathbf{S}$ et select the leading C eigenvectors of $\mathbf{S}$ as optimal projecting directions.

In this way we focus the attention on information that discriminate the characteristic forms of classes, *i.e.* target values.

4.3.2   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 2015, an answer to the questions above was given in [CK14b], linked to the concept of *explained variance* (or *explained global variance*, EGV for short) of a PC $\boldsymbol{\alpha}_i$:

$$\mathrm{EGV}(\boldsymbol{\alpha}_i) = \frac{\lambda_i}{\sum_{k=1}^{r} \lambda_k} \, , \tag{4.11}$$

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; for this reason this quantity is often multiplied by 100 and expressed as percentage. Exploiting the EGV to choose among the PCs consists in fixing a wished *cumulative explained variance* β and in keeping C different PCs, where C is the minimum integer such that

$$\mathrm{EGV}(\boldsymbol{\alpha}_1) + \mathrm{EGV}(\boldsymbol{\alpha}_2) + \cdots + \mathrm{EGV}(\boldsymbol{\alpha}_C) \geq \beta \, . \tag{4.12}$$

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

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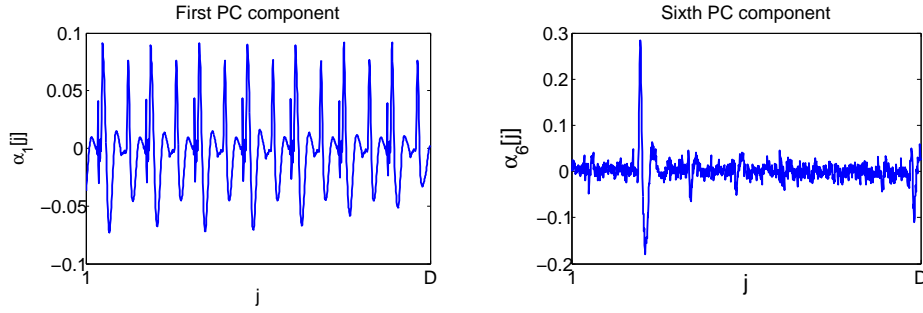
FIGURE 4.3: First and sixth PCs in DPA contest v4 trace set (between time samples 198001 and 199000)

(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:

$$\mathrm{IPR}(\boldsymbol{\alpha}_i) = \sum_{j=1}^{D} \boldsymbol{\alpha}_i[j]^4 \, . \qquad (4.13)$$

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.

4.3.3 The Explained Local Variance Selection Method

The method we develop in this section is based on a compromise between the variance provided by each PC (more precisely its EGV) and the number of time samples necessary to achieve a consistent part of such a variance. To this purpose we introduce the concept of *Explained Local Variance* (ELV). Let us start by giving some intuition behind our new concept. Thinking to the observations $\mathbf{x}^\intercal$, or to the class-averages $\overline{\mathbf{x}}^\intercal$ in class-oriented PCA case, as realizations of a random variable $\vec{X}^\intercal$, we
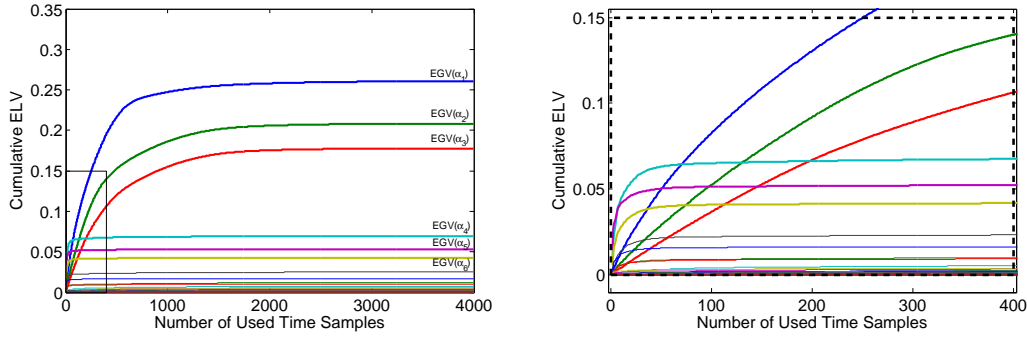
FIGURE 4.4: Cumulative ELV trend of principal components. On the
right a zoom of the plot on the left. Data acquisition described in
Sec. 4.5.

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

$$\lambda_i = \hat{\mathrm{Var}}(\sum_{j=1}^{D} \vec{X}^\intercal[j]\boldsymbol{\alpha}_i[j]) = \sum_{j=1}^{D}\sum_{k=1}^{D} \hat{\mathrm{Cov}}(\vec{X}^\intercal[j]\boldsymbol{\alpha}_i[j], \vec{X}^\intercal[k]\boldsymbol{\alpha}_i[k]) = \tag{4.14}$$

$$= \sum_{j=1}^{D} \boldsymbol{\alpha}_i[j] \sum_{k=1}^{D} \boldsymbol{\alpha}_i[k]\hat{\mathrm{Cov}}(\vec{X}^\intercal[j], \vec{X}^\intercal[k]) = \sum_{j=1}^{D} \boldsymbol{\alpha}_i[j](\mathbf{S}_j^\intercal \cdot \boldsymbol{\alpha}_i) = \tag{4.15}$$

$$= \sum_{j=1}^{D} \boldsymbol{\alpha}_i[j]\lambda_i\boldsymbol{\alpha}_i[j] = \sum_{j=1}^{D} \lambda_i\boldsymbol{\alpha}_i[j]^2 \tag{4.16}$$

where $\mathbf{S}_j^\intercal$ denotes the j-th row of $\mathbf{S}$ and (4.16) is justified by the fact that $\boldsymbol{\alpha}_i$ is an
eigenvector of $\mathbf{S}$, with λ_i its corresponding eigenvalue. The result of this computation is quite obvious, since $\| \boldsymbol{\alpha}_i \| = 1$, but it evidences the contribution of each time
sample in the information held by the PC. This makes us introduce the following
definition:

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

$$\mathrm{ELV}(\boldsymbol{\alpha}_i, j) = \frac{\lambda_i\boldsymbol{\alpha}_i[j]^2}{\sum_{k=1}^{r} \lambda_k} = \mathrm{EGV}(\boldsymbol{\alpha}_i)\boldsymbol{\alpha}_i[j]^2 \ . \tag{4.17}$$

Let $\mathcal{J} = \{j_1^i, j_2^i, \dots, j_D^i\} \subset \{1, 2, \dots, D\}$ 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 $j \in [1, \dots, D]$, 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. 4.4, where such cumulative ELVs are represented, the first 3
components are much slower in achieving their final EGV, while the 4th, the 5th and
the 6th 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 $k \in [1, \dots, 30]$, 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 4th, the 5th
and the 6th component only essentially depends on a very few time samples. This
observation, combined with Assumption 1, suggests that they are more suitable for
SCA than the three first ones. To validate this statement, it suffices to look at the
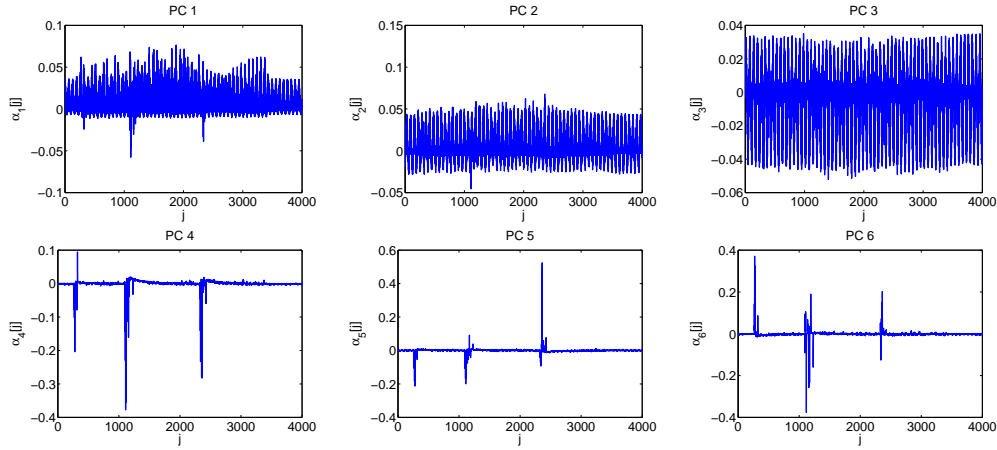
FIGURE 4.5: The first six PCs. Acquisition campaign on an 8-bit AVR Atmega328P (see Sec. 4.5).

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

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

We remark that the ELV is a score associated not only to the whole components, but to each of their coefficients. This interesting property can be exploited to further remove, within a selected PC, the non-significant points, *i.e.* those with a low ELV. In practice this is done by setting these points to zero. That is a natural way to exploit the ELV score in order to operate a kind of *denoising* for the reduced data, making them only depend on the significant time samples. In Sec. 4.5 (scenario 4) we test the performances of an attack varying the number of time samples involved in the computation of the reduced data, and showing that such a denoising processing might impact significantly.

4.4 Linear Discriminant Analysis

Fisher's Linear Discriminant and Terminology Remark

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

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

Description.

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

$$\boldsymbol{\alpha}_1 = \text{argmax}_{\boldsymbol{\alpha}} \frac{\boldsymbol{\alpha}^\intercal \mathbf{S_B} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\intercal \mathbf{S_W} \boldsymbol{\alpha}} \, , \tag{4.18}$$

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

$$\mathbf{S_W} = \sum_{z \in \mathcal{Z}} \sum_{i=1}^{N_p} (\mathbf{x}_i - \bar{\mathbf{x}}_z)(\mathbf{x}_i - \bar{\mathbf{x}}_z)^\intercal. \tag{4.19}$$

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

$$\mathbf{S} = \frac{1}{N_p}(\mathbf{S_W} + \mathbf{S_B}) \, . \tag{4.20}$$

It can be shown that the vector $\boldsymbol{\alpha}_1$ which maximizes (4.18) must satisfy $\mathbf{S_B}\boldsymbol{\alpha}_1 = \lambda \mathbf{S_W}\boldsymbol{\alpha}_1$, for a constant λ, *i.e.* has to be an eigenvector of $\mathbf{S_W}^{-1}\mathbf{S_B}$. Moreover, for any eigenvector $\boldsymbol{\alpha}$ of $\mathbf{S_W}^{-1}\mathbf{S_B}$, with associated eigenvalue λ, the Rayleigh quotient equals such a λ:

$$\frac{\boldsymbol{\alpha}^\intercal \mathbf{S_B} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\intercal \mathbf{S_W} \boldsymbol{\alpha}} = \lambda \, . \tag{4.21}$$

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

The computation of the eigenvectors of $\mathbf{S_W}^{-1}\mathbf{S_B}$ is known under the name of *generalized eigenvector problem*. The difficulty here comes from the fact that $\mathbf{S_W}^{-1}\mathbf{S_B}$ is not guaranteed to be symmetric. Due to this non-symmetry, $\boldsymbol{\alpha}_1$ and the others eigenvectors do not form an orthonormal basis for $\mathbb{R}^D$, but they are anyway useful for classifications scopes, as in SCA. Let us refer to them as *Linear Discriminant Components* (LDCs for short); as for PCs we consider them sorted in decreasing order with respect to their associated eigenvalue, which gives a score for their informativeness, see (4.21). Analogously to the PCA, the LDA provides a natural dimensionality reduction: one can project the data over the C first LDCs. As for PCA, this choice might not be optimal when applying this reduction to side-channel traces. For the sake of comparison, we tested all the selection methods proposed for the PCA (EGV, IPR and ELV) in association to the LDA as well.

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

4.4.1 The Small Sample Size Problem

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

If the singularity of $\mathbf{S_B}$ does not affect the LDA dimensionality reduction, we cannot say the same about the singularity of $\mathbf{S_W}$: SCA and Pattern Recognition literatures point out the same drawback of the LDA, known as the *Small Sample Size problem* (SSS for short). It occurs when the total number of acquisitions N_p is less than or equal to the size D of them.[1] The direct consequence of this problem is the singularity of $\mathbf{S_W}$ and the non-applicability of the LDA.

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

Fisherface Method

The most popular among the solutions to SSS is the so-called *Fisherface* method[2] [BHK97]. It simply relies on the combination between PCA and LDA: a standard PCA dimensionality reduction is performed to data, making them pass from dimension D to dimension $N_p - |\mathcal{Z}|$, which is the general maximal rank for $\mathbf{S_W}$. In this reduced space, $\mathbf{S_W}$ is very likely to be invertible and the LDA therefore applies.

$\mathbf{S_W}$ Null Space Method

It has been introduced by Chen et al. in [Che+00] and exploits an important result of Liu et al. [LCY92] who showed that Fisher's criterion (4.18) is equivalent to:

$$\boldsymbol{\alpha}_1 = \text{argmax}_{\boldsymbol{\alpha}} \frac{\boldsymbol{\alpha}^\intercal \mathbf{S_B} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\intercal \mathbf{S_W} \boldsymbol{\alpha} + \boldsymbol{\alpha}^\intercal \mathbf{S_B} \boldsymbol{\alpha}} \ . \tag{4.22}$$

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

---

[1] It can happen for example when attacking an RSA implementation, where the acquisitions are often huge (of the order of 1,000,000 points) and the number of measurements may be small when the SNR is good, implying that a good GE can be achieved with a small N.

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

Direct LDA

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

$\mathbf{S_T}$ Spanned Space Method

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

Remark. Let us remark that, from a computational complexity point of view (see [Hua+02] for a deeper discussion), the least time-consuming procedure among the four proposed is the Direct LDA, followed by the Fisherface and the $\mathbf{S_T}$ Spanned Space Method, that have a similar complexity. The $\mathbf{S_W}$ Null Space Method is in general much more expensive, because the task of removing the $\mathbf{S_W}$ null space requires the actual computation of the $(D \times D)$-dimensional matrix $\mathbf{S_W}$, *i.e.* the computational trick proposed by Archambeau et al. [Arc+06], see Sec. **??** is not applicable. On the contrary, the other three methods take advantage of such a procedure, reducing drastically their complexity.

4.5   Experimental Results

In this section we compare the different extractors (*i.e.* functions applying a data dimensionality reduction, see Sec. 2.1) provided by the PCA and the LDA in association with the different techniques of components selection. Defining an universal criterion to compare the different extractors would not make sense since the latter one should encompass a lot of parameters, sometimes opposite, that vary according to the context (amount of noise, specificity of the information leakage, nature of the side channel, etc.). For this reason we choose to split our comparisons into four scenarios. Each scenario has a single varying parameter that, depending on the attacker context, may wish to be minimized. Hereafter the definition of the four scenarios:

Scenario 1 varying parameter: number of attack traces N_a,

Scenario 2 varying parameter: number of profiling traces N_p,

Scenario 3 varying parameter: number of projecting components selected C,
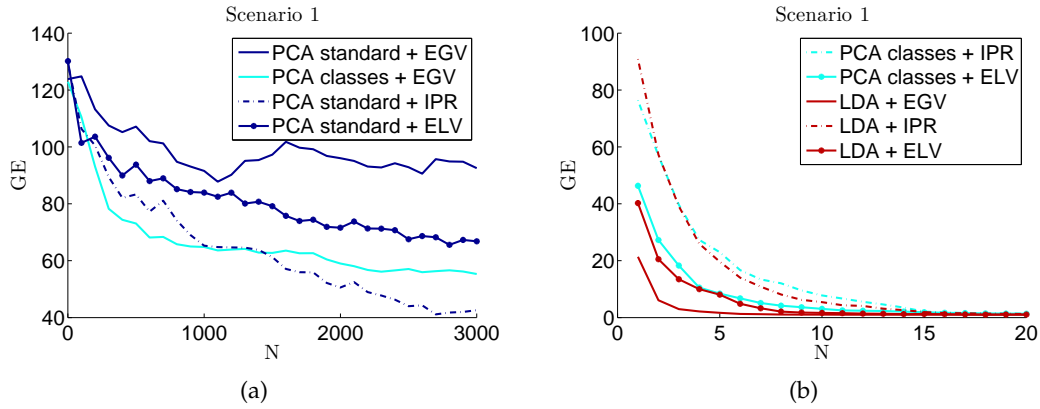
FIGURE 4.6: Guessing Entropy as function of the number of attack traces for different extraction methods. All Guessing Entropies are estimated as the average rank of the right key over 100 independent experiments.

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. 4.4.1 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. 4.3.2.

The testing adversary.

Our testing adversary attacks an 8-bit AVR microprocessor Atmega328P and acquires power-consumption traces via the ChipWhisperer platform [OC14].[3] The target device stores a secret 128-bit key and performs the first steps of an AES: the loading of 16 bytes of the plaintext, the AddRoundKey step and the AES Sbox. It has been programmed twice: two different keys are stored in the device memory during the acquisition of the profiling and of the attack traces, to simulate the situation of two identical devices storing a different secret. The size D of the traces equals 3996. The sensitive target variable is the output of the first Sbox processing, but, since the key is fixed also during the profiling phase, and both Xor and Sbox operations are bijective, we expect to detect three interesting regions (as those high-lighted by PCs 4, 5 and 6, in Fig. 4.5): the reading of the first byte of the plaintext, the first AddRound-Key and the first Sbox. We consider an *identity classification* leaking function (i.e. we make minimal assumption on the leakage function), which implies that the 256 possible values of the Sbox output yields to 256 classes. For each class we assume that the adversary acquires the same number N of traces, *i.e.* $N_p = N \times 256$. After the application of the extractor ϵ, the trace size is reduced to C. Then the attacker performs a Template Attack (see Sec. 2.1, using C-variate Gaussian templates.

---

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

Scenario 1.

To analyse the dependence between the extraction methods presented in Sections 4.2 and 4.4 and the number of attack traces N_a needed to achieve a given GE, we fixed the other parameters as follows: $N = 50$ ($N_p = 50 \times 256$), $C = 3$ and $\sharp \mathrm{PoI} = 3996$ (all points are allowed to participate in the building of the PCs and of the LDCs). The experimental results, depicted in Fig. 4.6(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, $\sharp \mathrm{PoI} = 3996$ again and the number of attack traces is $N_a = 100$. This scenario has to be divided into two parts: if $N_z \leq 15$, then $N_p < D$ and the SSS problem occurs. Thus, in this case we test the four extensions of LDA presented in Sec. 4.4.1, associated to either the standard selection, to which we abusively refer as EGV,[4] or to the IPR selection. We compare them to the class-oriented PCA associated to EGV, IPR or ELV. The ELV selection is not performed for the techniques extending LDA, since for some of them the projecting LDCs are not associated to some eigenvalues in a meaningful way. On the contrary, if $N \geq 16$ there is no need to approximate the LDA technique, so the classical one is performed. Results for this scenario are shown in Fig. 4.7. It may be noticed that the combinations class-oriented PCA + ELV/IPR select exactly the same components, for our data, see Fig. 4.7(e) and do not suffer from the lack of profiling traces. They are slightly outperformed by the $\mathbf{S_W}$ Null Space method associated with the EGV, see Fig.4.7(d). The Direct LDA (Fig. 4.7(b)) method also provides a good alternative, while the other tested methods do not show a stable behaviour. The results in absence of the SSS problem (Fig.4.7(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.

Scenario 3.

Let C be now variable and let the other parameters be fixed as follows: $N_a = 100, N_z = 200, \sharp \mathrm{PoI} = 3996$. Looking at Fig. 4.8, 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. 4.7(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.

---

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

FIGURE 4.7: Guessing Entropy as function of the number of profiling traces. Figures (a)-(d): methods extending the LDA in presence of SSS problem; Figure (e): class-oriented PCA in presence of the SSS problem; Figure (f): number of profiling traces high enough to avoid the SSS problem.
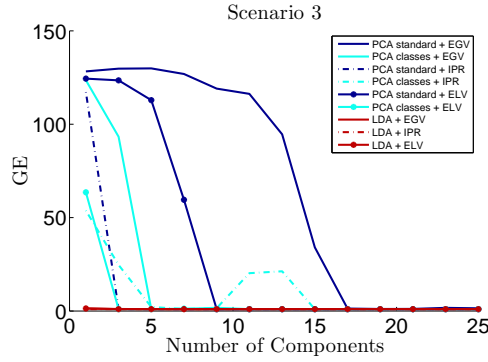
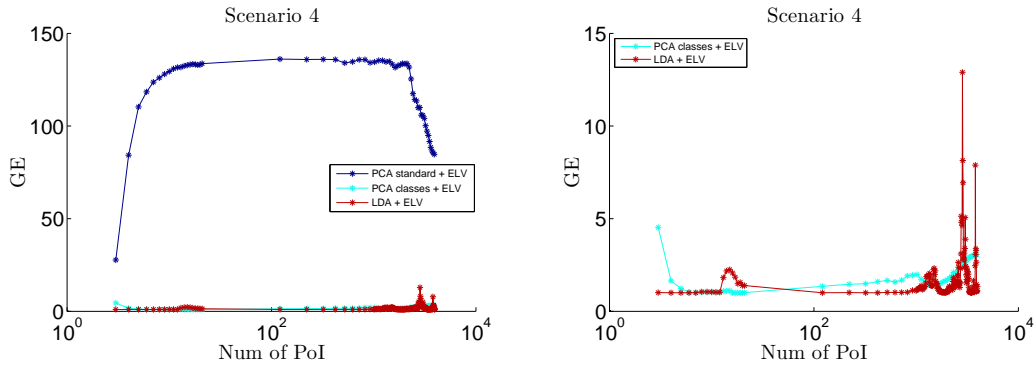FIGURE 4.8: Guessing Entropy as function of the trace size after reduction.

FIGURE 4.9: Guessing Entropy as function of the number of time samples contributing to the extractor computation.

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 $\sharp$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 $\sharp$PoI non-zero columns. Looking at Fig. 4.9 we might observe that the LDA allows to achieve the maximal guessing entropy with only 1 PoI in each of the 3 selected components. Actually, adding PoIs worsen its performances, which is coherent with the assumption that the vulnerable information leaks in only a few points. Such points are excellently detected by the LDA. Adding contribution from other points raises the noise, which is then compensated by the contributions of further noisy points, in a very delicate balance. Such a behaviour is clearly visible in standard PCA case: the first 10 points considered raise the level of noise, that is then balanced by the last 1000 points.

4.5.1 Overview of this Study and Conclusions

This study focused two well-known techniques to construct extractors for side-channel traces, the PCA and the LDA. The LDA method is more adequate than the PCA one, thanks to its class-distinguishing asset, but more expensive and not always available in concrete situations. We deduced from a general consideration about side-channel

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

TABLE 4.1: Overview of extractors performances in tested situations.

traces, *i.e.* the fact that for secured implementations the vulnerable leakages are concentrated into few points, a new methodology for selecting components, called ELV. We showed that the class-oriented PCA, equipped with the ELV, achieves performances close to those of the LDA, becoming a cheaper and valid alternative to the LDA. Being our core consideration very general in side-channel context, we believe that our results are not case-specific. The ELV tool has been tested among others in a posterior study [CK18]: in this case it did not ameliorate the attack performances, neither made them worse, since it performed the same component selection as the EGV.

A second part of the proposed study analysed experimentally some alternatives to the LDA in presence of SSS problem proposed in Pattern Recognition literature. Such experiments showed that the Direct LDA and the $\mathbf{S_W}$ Null Space Method are promising techniques, exhibiting performances close to the ones given by the ELV-equipped class-oriented PCA.

4.6 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. 4.5, and artificially misalign them through the technique proposed in Appendix A 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. 4.10 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.
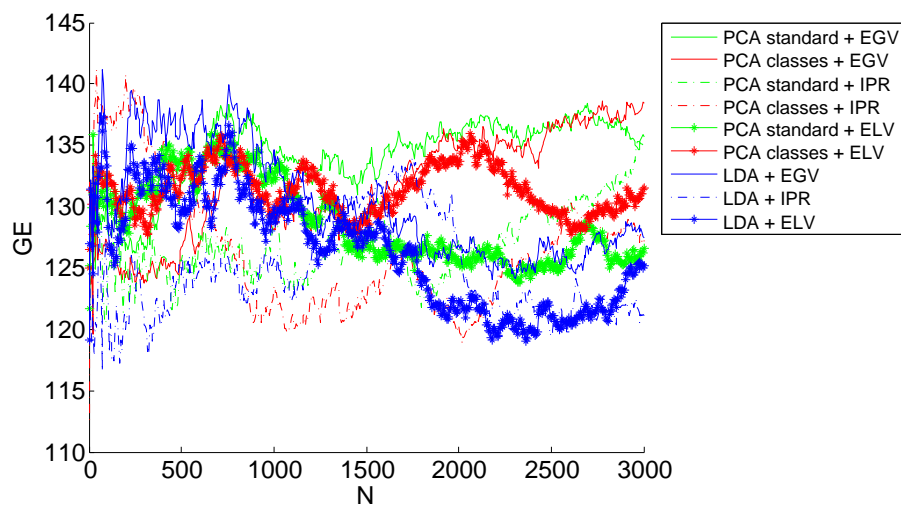
FIGURE 4.10: Degradation of linear-reduction-based template attacks
in presence of misalignment.

Chapter 5

Kernel Discriminant Analysis

We tackle the dimensionality reduction problem in the context of profiling attacks against implementations protected by masking countermeasure. For such attacks, the attacker might have or not access to the random values drawn at every execution and used to mask the sensitive variables. If he have such a knowledge, then the dimensionality reduction problem turns to be equivalent to the case of unprotected implementations. Thus ,the classic statistics for the PoIs research (and we will concentrate over the SNR, see Sec. 2.1) and the linear dimensionality reduction techniques described in the previous chapter are still applicable and efficient. On the contrary, when such a knowledge is denied, linear techniques are completely inefficient: a non-linear function of the PoIs has to be considered in order to construct discriminant features from side-channel observations. In this chapter we propose to make use of Kernel Discriminant Analysis (KDA) technique to construct such a non-linear processing. To this aim we revisit the contents and the experimental results of the paper presented at CARDIS 2016 international conference [CDP16b]. After such a publication, the KDA has been compared to other non-linear dimensionality reduction techniques in [Ou+17], where manifold learning solutions such as ISOMAP, Locally Linear Embedding (LLE) and Laplacian Eigenmaps (LE) are proposed. Moreover, a use of the KDA in an unsupervised way to perform a higher-order SCA (as a key candidate distinguisher and not as a dimensionality reduction technique) has been proposed at CARDIS 2017 [Zho+17].

5.1 Motivation

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

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

5.1.1   Getting information from masked implementations

The SNR measures, point by point, the information held by the first-order moment of the acquisition, *i.e.* the mean, to which we can refer to as a *1st-order information*. In masked implementations, such information is null: in any time sample the mean is independent from Z due to the randomization provided by the shares, namely $f(z) = \mathbb{E}[\vec{X}|Z = z]$ is constant, which implies that the SNR is asymptotically null over the whole trace.

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

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

Remark. The use of central moments has been experimentally shown to reveal more information than the use of the raw ones [**ProuffRB09**; Cha+99]. Thus we will from now on suppose that the acquisitions have previously been normalized, so that $\hat{\mathbb{E}}(\mathbf{x}_i) = \mathbf{0}$ and $\hat{\mathrm{Var}}(\mathbf{x}_i) = \mathbf{1}$. In this way a centred product coincides with a non-centred one.

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

5.1.2   Some strategies to perform higher-order attack

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

---

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

[2] not necessary distinct

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

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

The third approach, whose complexity does not increase exponentially with d, is the application of the higher-order version of the PP tool for the PoI selection, for which we give an outline hereafter. As will be discussed in Sec. 5.4.4, its heuristic nature is the counterpart of the relatively restrained complexity of this tool.

Higher-Order Version of Projection Pursuits

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

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

where $\boldsymbol{\alpha}$ is a sparse projecting vector with d non-overlapping windows of coordinates set to 1, where the method has identified points of interest. Actually, as will be discussed in Sec. 5.4.4, authors of [Dur+15] propose to exploit $\boldsymbol{\alpha}$ as a pointer of PoIs, but do not encourage the use of ϵ^{PP} as an attack preprocessing. The procedure is divided into two parts: a global research called *Find Solution* and a local optimization called *Improve Solution*. At each step of *Find Solution*, d windows are randomly selected to form a primordial $\boldsymbol{\alpha}$, thus a primordial ϵ^{PP}. A part of the training traces are then processed via ϵ^{PP} and used to estimate the dth-order statistical moments $\mathbf{m}_Z^d = \hat{\mathbb{E}}_i[(\epsilon^{PP}(\mathbf{x}_i))_{i:z_i=z}]$, for each value of Z. Then the Pearson correlation coefficient $\hat{\rho}$ between such estimates and the same estimates issued from a second part of the training set is computed. If $\hat{\rho}$ is higher than some threshold T_{det}, the windows forming $\boldsymbol{\alpha}$ are considered interesting[3] and *Improve Solution* optimises their positions and lengths, via small local movements. Otherwise, the $\boldsymbol{\alpha}$ is discarded and another d-tuple of random windows is selected.

The threshold T_{det} plays a fundamental role in this crucial part of the algorithm: it has to be small enough to promote interesting windows (avoiding false negatives)

---

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

$$\mathbb{R}^D \xrightarrow{\quad \Phi \quad} \mathcal{F} \xrightarrow[\epsilon^{\mathrm{LDA}}]{\epsilon^{\mathrm{PCA}}} \mathbb{R}^C$$

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

and high enough to reject uninteresting ones (avoiding false positives). A hypothesis test is used to choose a value for T_{det} in such a way that the probability of $\hat{\rho}$ being higher than T_{det} given that no interesting windows are selected is lower than a chosen significance level β.[4]

5.1.3 Foreword of this study

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

5.2 Feature Space, Kernel Function and Kernel Trick

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

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

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

---

[4]Interestingly, the threshold T_{det} depends on size of $\mathcal{Z}$ and not on the size of the training sets of traces. This fact disables the classic strategy that consists in enlarging the sample, making T_{det} lower, in order to raise the statistical power of the test (*i.e.* $\mathrm{Prob}[\hat{\rho} > T_{det}|\rho = 1]$). Some developments of this algorithm have been proposed [DS15], also including the substitution of the MMPC objective function with a *Moments against Samples* one, that would let T_{det} decrease when increasing the size of the training set.
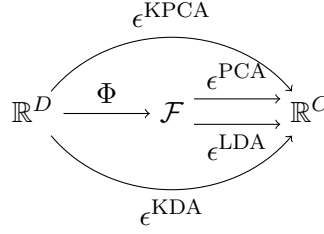
FIGURE 5.2: Applying KDA and KPCA permits to by-pass computations in $\mathcal{F}$.

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

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

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

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

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

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

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

$$K \colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$$
$$K \colon (\mathbf{x}_i, \mathbf{x}_j) \mapsto (\mathbf{x}_i \cdot \mathbf{x}_j)^2 \,, \tag{5.4}$$

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

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

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

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

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

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

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

5.2.1 Local Kernel Functions as Similarity Metrics

give an intuition and anticipate siamese networks

5.3 Kernel Discriminant Analysis

As shown in 5.3, a kernel function K allows us to compute the dot product between elements mapped into the feature space $\mathcal{F}$ (5.3), and more generally any procedure that is only composed of such products. Starting from this remark, the authors of [SSM98] have shown that the PCA and LDA procedures can be adapted to satisfy the latter condition, which led them to define the KPCA and KDA algorithms. The latter one is described in the following procedure. The interested reader will find the formal derivation of KPCA and the application of the same kernel trick to the class-oriented version of PCA in Appendix B. It is given as a way of example of how one can translate the PCA algorithm in such a way of never access data but only dot product between data.

Procedure 1 (KDA for dth-order masked side-channel traces). Given a set of labelled profiling side-channel traces $(\mathbf{x}_i, z_i)_{i=1,\dots,N_p}$ and the kernel function $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$:

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

$$\mathbf{M} = \sum_{z \in \mathcal{Z}} N_z (\mathbf{M}_z - \mathbf{M_T})(\mathbf{M}_z - \mathbf{M_T})^\mathsf{T} , \qquad (5.7)$$

where $\mathbf{M}_z$ and $\mathbf{M_T}$ are two N-size column vectors whose entries are given by:

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

$$\mathbf{M_T}[j] = \frac{1}{N_p} \sum_{i=1}^{N_p} K(\mathbf{x}_j, \mathbf{x}_i) . \qquad (5.9)$$

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

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

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

---

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

2bis) Regularize the matrix $\mathbf{N}$ for computational stability:

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

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

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

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

For the reasons discussed in the introduction of this section, the right-hand side of (B.25) may be viewed as an efficient way to process the ℓ-coordinate of the vector $\epsilon^{LDA}(\Phi(\mathbf{x}) = \mathbf{w}_\ell \cdot \Phi(\mathbf{x})$, without evaluating $\Phi(\mathbf{x})$. The entries of $\mathbf{w}_\ell$ are never computed, and will thus be referred to as *implicit coefficients* (see Remark 5.3). It may be observed that each discriminant component $\epsilon_\ell^{\text{KDA}}(\cdot)$ depends on the training set $(\mathbf{x}_i, z_i)_{i=1,\ldots,N_p}$, on the kernel function K and on the regularization parameters μ.

Remark (The implicit coefficients). As already said, the KDA, when the dth-degree polynomial kernel function is chosen as kernel function, operates implicitly in the feature space of all products of d-tuples of time samples. In order to investigate the effect of projecting a new trace $\mathbf{x}$ over a component $\epsilon_\ell^{\text{KDA}}(\mathbf{x})$, we can compute for a small d the implicit coefficients that are assigned to the d-tuples of time samples through (B.25). For $d = 2$ we obtain that in such a feature space the projection is given by the linear combination computed via the coefficients shown below:

$$\epsilon_\ell^{\text{KDA}}(\mathbf{x}) = \sum_{j=1}^{D}\sum_{k=1}^{D} [(\mathbf{x}[j]\mathbf{x}[k]) \underbrace{(\sum_{i=1}^{N_p} \boldsymbol{\nu}_\ell[i]\mathbf{x}_i[j]\mathbf{x}_i[k])}_{\text{implicitcoefficients}}] \tag{5.13}$$

5.3.1 Computational complexity analysis

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

- $\frac{N_p^2}{2}D$ multiplications, $\frac{N_p^2}{2}(D-1)$ additions and $\frac{N_p^2}{2}D$ raising to the d-th power, to process the kernel function over all pairs of training traces

- $(D+C)$ multiplications, $(D+C-2)$ additions and 1 raising to the d-th power for the projection of each new trace over C KDA components,

- the cost of the eigenvalue problem, that is $O(N_p^3)$.

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

5.4 Experiments over Atmega328P

5.4.1 Experimental Setup

The target device is an 8-bit AVR microprocessor Atmega328P and we acquired power-consumption traces thanks to the ChipWhisperer platform [OC14].[6] From the acquisitions we extracted some traces composed of 200 time samples, corresponding to 4 clock cycles (see Fig.5.7(a) or 5.7(b) upper parts). Depending on the attack implementation, we ensure that the acquisitions contain either 2,3 or 4 shares respectively for $d = 2, 3$ or 4. The shares satisfy $M_1 \oplus \cdots \oplus M_d = Z$, where $Z = \text{Sbox}(P \oplus K^\star)$. The goal of the attack is to recover the subkey $K^\star$. The plaintext P is assumed to be known and the M_i are assumed to be unknown random uniform values. For the characterization phase, the subkey $K^\star$ is also known. The training set has size $N_p = 8960$, and the plaintexts p_i have been chosen to balance the number of classes ($N_z = \frac{8960}{256} = 35$ for each $z \in \mathbb{F}_2^8$). We fixed the dimension C at the value 2 (except for the 2-class KDA for which we chose $C = 1$, see Remark 5.4.3): we therefore tried to build extractors $\epsilon^{\text{KDA}}(\cdot) = (\epsilon_1^{\text{KDA}}(\cdot), \epsilon_2^{\text{KDA}}(\cdot))$ mapping traces of size 200 samples into new traces composed of 2 coordinates.[7] Afterwards, a bivariate Gaussian TA (see Sec. 2.1).

As discussed in Remark 5.1.1, the training traces are normalized. The average trace and the standard deviation trace used to perform the normalization are stored and reused to center the profiling and attack traces before projecting them onto the KDA components. In this way the new traces present a form as similar as possible to the training ones. Remark that, due to the overfitting risk, discussed in general in Sec. 3.1.4 and that will be discussed in the particular case of KDA in Sec. 5.4.2, it is worth to not reusing the training traces as profiling traces in the attack phase: the extractor ϵ^{KDA} has a different behaviour when applied to the traces used to train it and to some new traces, and this might corrupt the profiling phase.

5.4.2 The Regularization Problem

By construction the matrix $\mathbf{N}$ in (5.10) is not positive-definite, which is one of the reasons why [SM99] proposes the regularization (5.11) recalled hereafter:

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

When applying such a regularization, the choice of the constant μ is crucial. For sure it has to be large enough to ensure that $\mathbf{N}$ turns to a positive-definite matrix, but we experimentally observed that the minimal μ for which the positive-definitiveness of $\mathbf{N}$ is attained is often far from being the one that provides a good extractor. In Fig. 5.3 (left) we observe the efficiency of a template attack run in combination with a KDA extractor. The matrix $\mathbf{N}$ is positive-definite for $\mu_1 = 10^{-3}$ but the value that provides the best extractor is much higher (namely $\mu_3 = 10^7$). Still raising the value of μ degrades the quality of the extractor. The right part of Fig. 5.3 shows the implicit coefficients of the extractor (see (5.13)) obtained under μ_3 (upper triangular part)

---

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

[7]As we have seen in Chapter 4, for PCA and LDA methods a good component selection is fundamental to obtain an efficient subspace, and that the first components not always represent the best choice. This is likely to be the case for the KDA as well, but in our experiments the choice of the two first components $\epsilon_1^{\text{KDA}}, \epsilon_2^{\text{KDA}}$ turns out to be satisfying, and therefore to simplify our study we preferred to not investigate other choices.
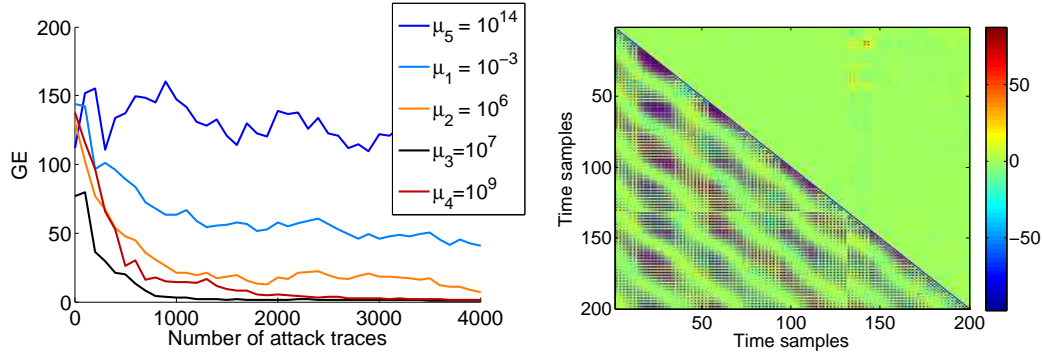
FIGURE 5.3: On the left: template attack guessing entropy vs the number of traces for the attack phase, varying for choices of the constant μ (5.11). On the right: the implicit coefficients assigned to pairs of time samples for μ_3 (upper triangular part) and μ_5 (lower triangular part).

and under μ_5 (lower triangular part). The extractor corresponding to the former one leads to a successful attack and has high values concentrated over the interesting windows; the extractor corresponding to the latter one leads to an unsuccessful attack and shows lack of localization around interesting parts of the trace, highlighting the fact that the KDA tool failed in detecting generalisable class-distinguishing features in this case.

The regularization (5.11) is a proper regularization in the sense discussed in Sec. 3.1.4: it is not only a way to make the problem computationally stable (which explains why the minimal μ making **N** positive-definite may not be a good choice), but also a answer to the overfitting phenomenon. In the case of the KDA the overfitting is observable when ϵ^{KDA} almost perfectly separates the training traces in their classes, while failing in separating the attack ones. As explained in [SM99], the regularization (5.11) corresponds to the additional requirement for $\boldsymbol{\nu}$ to have a small norm $\|\boldsymbol{\nu}\|^2$. As any regularization technique, it makes the method less accurate in the learning phase, but in some cases more likely to correctly operate on new examples.

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

Some heuristics exist to choose the constant μ; *e.g.* the average of the diagonal entries [LZO06] or the minimal constant that let **N** be diagonally dominant (implying positive-definite), etc. In [CL06] Centeno *et al.* propose a maximization strategy to find the optimal regularization parameter, based on a probabilistic approach. The study of this approach is left for future works. For our study we simply performed experiments for various values of μ and we kept the best obtained results.

Remark. A criterion to decide whether the chosen regularization is convenient consists in comparing the SNR computed over the extracted training traces $(\epsilon^{\mathrm{KDA}}(\mathbf{x}_i))_{i=1,\ldots,N_p}$ with the one computed over some extracted new traces $\mathbf{y}_i$ (*e.g.* the profiling traces $(\epsilon^{\mathrm{KDA}}(\mathbf{y}_i))_{i=1,\ldots,N_p}$). If the latter is significantly lower than the former, the regularization failed and it is worth to try with another constantμ.
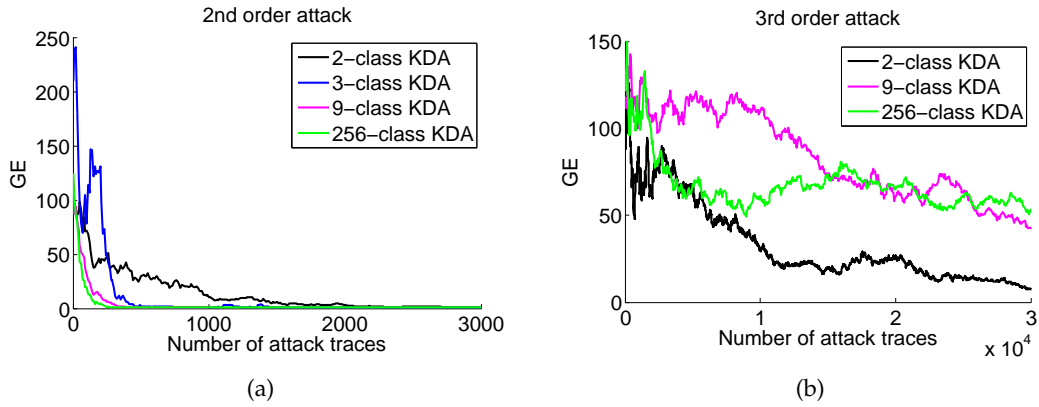
FIGURE 5.4: Comparison between 2-class,3-class, 9-class and 256-class KDA in 2nd-order context (a) and in 3rd-order context (b). For 2nd-order the KDA is efficient in providing separability between 256 classes, allowing an optimal attack. In 3rd-order context the training data are not enough to succeed the 256-class learning phase. Decreasing the number of classes to be distinguished raises the efficiency of the learning problem and thus of the attack.

5.4.3   The Multi-Class Trade-Off

As discussed in Sec. 4.4, the LDA, and by consequence the KDA, looks for a subspace of the feature space to optimally separate some given classes (for a well-chosen distance). Intuitively the more examples(*i.e.* training traces) available the more accurate the resulting KDA. Nevertheless, the number of examples might be bounded by the acquisition context. Actually even when the number N_p can be very high, it may be interesting to minimize it since the KDA complexity is $O(N_p^3)$. A trade-off must therefore be found between accuracy and efficiency. Assuming that the size of the training set is fixed to N_p, which controls the efficiency, a way to gain in accuracy may be found by appropriately adjusting the number of classes to distinguish: again intuitively, the more examples per class, the more accurate the detection of a separating subspace. Then, if the total number of training traces is fixed, in order to raise the number of traces per class, a smaller number of classes must be considered. To do so, a non-injective model $m(\cdot)$ can be introduced, to create a smaller set of labels $m(\mathcal{Z})$ from the initial set $\mathcal{Z}$. In the rest of the paper, the reduced number of classes, *i.e.* the number of labels assigned to the training set after applying the model m, will be denoted by W (it is the cardinality of $m(\mathcal{Z})$). In order to not corrupt the soundness of the KDA method in the SCA context, it is worth choosing a meaningful model. As discussed in Sec. 2.1, a widely-accepted power-consumption model for side-channel traces is provided by the Hamming Weight (HW) function, it makes sense to group the different values of Z according to their Hamming weights. In Fig. 5.4 we compared a 2-class model ($m(z) = 0$ if $\text{HW}(z) < 4$, $m(z) = 1$ if $\text{HW}(z) \geq 4$), a 3-class model ($m(z) = 0$ if $\text{HW}(z) < 4$, $m(z) = 1$ if $\text{HW}(z) = 4$, $m(z) = 2$ if $\text{HW}(z) > 4$) and a 9-class one ($m(z) = \text{HW}(z)$).

Remark. A balanced training set of size $N_p = 9000$ (instead of 8960) has been used to run the experiments for 2-class, 3-class and 9-class KDA.

Remark. For the sake of consistency[8] between the pre-processing phase and the attack phase, when a non-injective model is applied to the labels of the training set

---

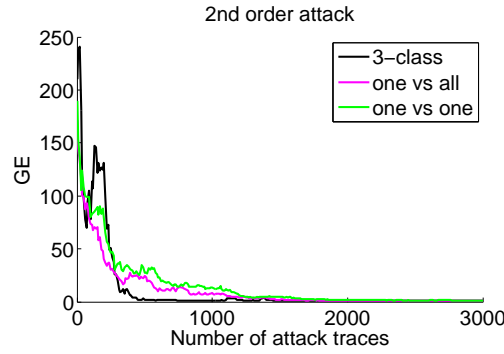[8]A different approach is analysed in Sec. 5.4.4.

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

to reduce the number of classes, the same model is exploited to run the template attack: W templates (one per each class) are computed and compared to the matching traces. Thus, in Fig. 5.4 the different multi-class extractors are compared using different template attacks. It may be remarked that as W decreases the efficiency of the attack is supposed to decrease as well, because each attack trace contributes in distinguishing the right key $K^\star$ only from a growing-size set of indistinguishable hypotheses.

In 2nd-order context, it can be observed in Fig. 5.4 that the KDA is provided with sufficient training traces to succeed a 256-class separation, which allows the finest characterization of the leakage, and leads as expected (see Remark 5.4.3), to the most efficient template attack. Moving to the 3-rd order context, the available training set is insufficient to make the multi-class approach succeed; nevertheless, turning the problem into a 2-class problem turns out to be a good strategy to trade extraction accuracy for attack efficiency.

Remark. The separating subspace given by the KDA has maximal dimension $(W-1)$, *i.e.* $Q \leq (W-1)$ in point 4 of Procedure 1. When $W = 2$ a single discriminant component $\epsilon_1^{\mathrm{KDA}}$ is available, then we run a univariate template attack, instead of a bivariate one.

An idea to avoid an excessive reduction of the number of separable classes W can be found in the machine learning literature dealing with the so-called *multi-class classification problems*. It consists in treating the W-class problem as multiple 2-class problems. Two different *modus operandi* exist: the *one-vs-one* and the *one-vs-all*. When applied to our context, the one-vs-one approach determines for each pair of classes the 1-dimensional subspace that best separates them and exploits all the obtained subspaces to run an attack (for W classes we obtain $\binom{W}{2}$ dimensions and we run a $\binom{W}{2}$-variate template attack). The one-vs-all approach looks for dimensions that best separate each class from all the others (obtaining W projections in total).

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

---

[9]We think that is result is quite data-dependant, so the use of such an approach is not discouraged in general.
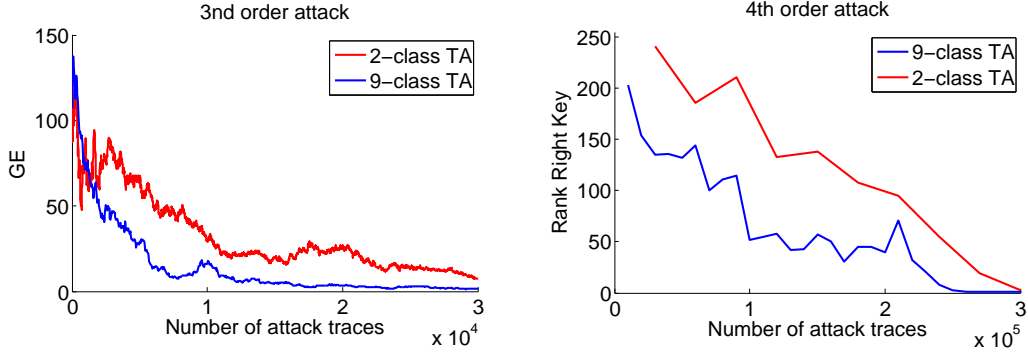
FIGURE 5.6: Left: guessing entropy (over 10 independent tests) for a
2-class and a 9-class 3rd-order template attack. Right: right key rank
of a 2-class and a 9-class 4th-order template attack.

our higher-order experiments.

5.4.4 Asymmetric Preprocessing/Attack Approach

In Remark 5.4.3 we appealed a consistency principle to justify the choice of run-
ning a W-class template attack after a W-class KDA extraction. Here we propose a
different reasoning: the consistency principle does not grant that an extractor ϵ^{KDA}
trained with W classes is not able to separate W' classes with $W' > W$. As seen
in Remark 5.3, an extractor ϵ^{KDA} always implicitly performs a weighed sum, via
the implicit coefficients, of centred products of time samples. If ϵ^{KDA} is effective,
the implicit coefficients which have the highest magnitude must correspond to time
samples corresponding to the manipulation of sensitive data (*e.g.* the variable shares
when masking is applied). This property is not necessarily related to the number of
classes used to train the the extractor.

Based on the reasoning above and on the observations reported in Remark 5.4.3,
we performed in 3rd-order and 4th-order contexts a 9-class template attack after
the 2-class KDA preprocessing. The results are depicted in Fig. 5.6 and confirm
that, for our experimental data, this approach is sound: in both cases, using the
same extractor trained with 2 classes and the same attack traces, the 9-class approach
outperforms the 2-class one.

Remark. We choose to run the 9-class instead of the 256-class template attack in order
to raise the accuracy of the profiling phase, being limited in terms of profiling traces.

Comparison with Projection Pursuits

To get a fair comparison, we run the PP algorithm (see Sec. 5.1.2) over the same train-
ing set used to evaluate the KDA in Sec.5.4. The best results in the 2nd-order context
were obtained with the HW model (*i.e.* $|\mathcal{Z}| = 9$). In this case T_{det} is fixed to 0.7.
Since 4 training sets are required, the 9000 training traces are split in 4 equally-sized
groups. Experimental observations allowed to fix $W_{len} = 5$, consequently suggest-
ing $minWS = 1$, $maxWS = 15$ and consistent global and local movements and
resizes. Given the heuristic asset of the algorithm, we run it 1000 times for $d = 2$ and
for $d = 3$. An overview of the global behaviour of the obtained results is depicted in
Figs 5.7(a) and 5.7(b): the lower parts of the figures show the sum of the 1000 out-
puts of the algorithm. We recall that each coordinate of $\boldsymbol{\alpha}$ is set to 1 for the windows

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

identified to be of interest, and to 0 elsewhere, so for each time sample the sum of the values (0 or 1) assigned by the 1000 attempts give an intuition about its likelihood to be considered as interesting by the PP method. It can be observed that in the 2-nd order case (Fig. 5.7(a)) the results are excellent: 100% of the tests highlight an informative part of the two clock-cycles where the sensitive shares are manipulated.[10] It means that $\epsilon^{PP}(\vec{X})$ always contains information about Z and a successful attack can be mounted over such extracted traces. The efficiency of such an attack depending on many factors, there is no interest in comparing it to the performances of the template attacks run in 2nd-order context using ϵ^{KDA} and depicted in Fig. 5.4(a). As it may be observed in Fig. 5.7(b), in the 3-rd order case the experimental results are completely different: almost no α selects the clock-cycle where the second share is manipulated. Thus in this case the PP approach fails: $\epsilon^{PP}(\vec{X})$ does not contain information about Z, so any attack launched over the extracted traces would fail, while ϵ^{KDA} still allows successful attacks in 3rd-order and 4th-order case, as depicted in Fig. 5.6.

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

5.5 Drawbacks of Kernel Methods

Misalignment Effects

Memory Complexity and Actual Number of Parameters

Two-Phases Approach: Preprocessing-Templates

---

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

Chapter 6

Convolutional Neural Networks against Jitter-Based Countermeasures

6.1 Moving from Kernel Machines to Neural Networks

6.2 Misalignment of Side-Channel Traces

6.2.1 The Necessity and the Risks of Applying Realignment Techniques

6.2.2 Analogy with Image Recognition Issues

6.3 Convolutional Layers to Impose Shift-Invariance

6.4 Data Augmentation for Misaligned Side-Channel Traces

cita o qui o all'inizio il paper di CARDIS 2017 sulla trace augmentation

6.5 Experiments against Software Countermeasures

6.6 Experiments against Artificial Hardware Countermeasures

6.7 Experiments against Real-Case Hardware Countermeasures

1

Chapter 7

Siamese Neural Networks for Collision Attacks

7.1 Introduction

7.2 Siamese Neural Networks

7.2.1 Distances and Loss Functions

7.2.2 Relation with Kernel Machines

7.3 Collision Attacks with Siamese NNs

7.3.1 Experimental Results

Chapter 8

Conclusions and Perspectives

8.1 Summary

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

Appendix A

Artificially Simulate Jitter

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

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

This deformation is applied to each clock pattern independently. We remark that is implies that, for example, The 19th clock cycle of a deformed acquisition suffers
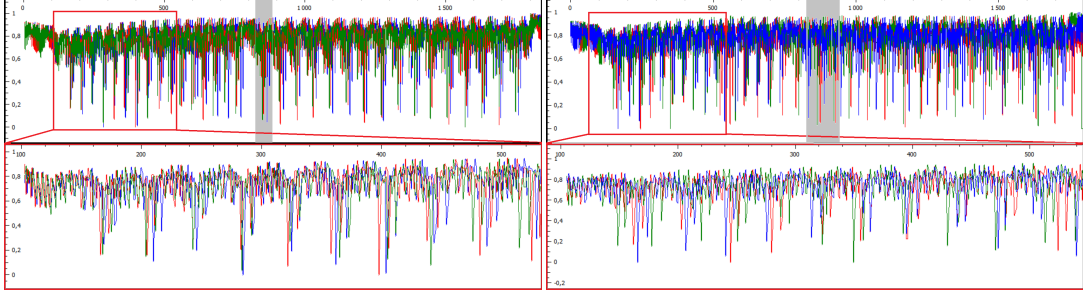
FIGURE A.1: Left: some traces of the *DS_low_jitter* dataset, a zoom of the part highlighted by the red rectangle is given in the bottom part. Right: some traces (and the relative) of the *DS_high_jitter* dataset. The interesting clock cycle for experiments in Sec. .6.6 is highlighted by the grey rectangular area.

from the cumulation of the previous 18 deformations. For the sake of visualizing the effect of such a jitter simulation, in Fig. A.1 we depict some traces of *DS_low_jitter* and of the *DS_high_jitter* datasets, used for experiments in Sec .6.6. They are obtained by perfectly synchronous acquisitions, with parameters set to `sigma` $= 2$, `B`$= 2$ for the *DS_low_jitter* dataset and `sigma` $= 6$, `B`$= 6$ for the *DS_high_jitter* one.

Appendix B

Kernel PCA construction

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

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

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

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

$$\mathbf{S}^\Phi \boldsymbol{\alpha}^\Phi = \lambda^\Phi \boldsymbol{\alpha}^\Phi \ . \tag{B.2}$$

We remark that such an eigenvector satisfies

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

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

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

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

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

---

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

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

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

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

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

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

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

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

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

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

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

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

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

which equals the following equation:

$$
N_p \lambda^{\Phi} \mathbf{K}\boldsymbol{\nu} = \mathbf{K}^2 \boldsymbol{\nu} .
\tag{B.15}
$$

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

$$
\gamma \boldsymbol{\nu} = \mathbf{K}\boldsymbol{\nu} .
\tag{B.16}
$$

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

$$\boldsymbol{\alpha}_k^{\Phi} \cdot \boldsymbol{\alpha}_k^{\Phi} = 1 \, , \tag{B.17}$$

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

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

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

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

B.1   Kernel class-oriented PCA

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

$$\mathbf{S_B}^{\Phi} \boldsymbol{\alpha}^{\Phi} = \lambda^{\Phi} \boldsymbol{\alpha}^{\Phi} \, , \tag{B.20}$$

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

$$\mathbf{S_B}^{\Phi} = \sum_{z \in \mathcal{Z}} N_z (\overline{\Phi(\mathbf{x})}^z - \overline{\Phi(\mathbf{x})}) (\overline{\Phi(\mathbf{x})}^z - \overline{\Phi(\mathbf{x})})^{\mathsf{T}} \, . \tag{B.21}$$

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

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

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

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

$$\gamma \mathbf{M} = \mathbf{M} \boldsymbol{\nu} \, , \tag{B.23}$$

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

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

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

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

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

Appendix C

Cross-Validation

Bibliography

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. "An implementation of DES and AES, secure against some attacks". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2001, pp. 309–318 (cit. on p. 28).

[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. 43, 44, 54).

[Bat+11] Lejla Batina et al. "Mutual information analysis: a comprehensive study". In: *Journal of Cryptology* 24.2 (2011), pp. 269–291 (cit. on pp. 23, 29).

[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. 21).

[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. 21).

[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. 23).

[BDP10] Martin Bär, Hermann Drexler, and Jürgen Pulkus. "Improved template attacks". In: *COSADE2010* (2010) (cit. on p. 26).

[BGK04] Johannes Blömer, Jorge Guajardo, and Volker Krummel. "Provably secure masking of AES". In: *International Workshop on Selected Areas in Cryptography*. Springer. 2004, pp. 69–83 (cit. on p. 28).

[BHK97] Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. 1997 (cit. on pp. 45, 53).

[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. 43, 44, 48).

[Bil+14]   Begül Bilgin et al. "A more efficient AES threshold implementation". In: *International Conference on Cryptology in Africa*. Springer. 2014, pp. 267–284 (cit. on p. 28).

[Bis06]   Christopher M.. Bishop. *Pattern recognition and machine learning*. Springer, 2006 (cit. on pp. 23, 24, 33, 45).

[BLR13]   Timo Bartkewitz and Kerstin Lemke-Rust. "Efficient Template Attacks Based on Probabilistic Multi-class Support Vector Machines". English. In: *Smart Card Research and Advanced Applications*. Ed. by Stefan Mangard. Vol. 7771. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 263–276. ISBN: 978-3-642-37287-2. DOI: `10.1007/978-3-642-37288-9_18`. URL: `http://dx.doi.org/10.1007/978-3-642-37288-9_18` (cit. on p. 39).

[Bru+ a]   N. Bruneau et al. "Less is more: Dimensionality reduction from a theoretical perspective". In: *17th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2015)*. to appear, 2015 (cit. on pp. 43, 51).

[Bru+14]   Nicolas Bruneau et al. "Boosting Higher-Order Correlation Attacks by Dimensionality Reduction". English. In: *Security, Privacy, and Applied Cryptography Engineering*. Ed. by RajatSubhra Chakraborty, Vashek Matyas, and Patrick Schaumont. Vol. 8804. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 183–200. ISBN: 978-3-319-12059-1. DOI: `10.1007/978-3-319-12060-7_13`. URL: `http://dx.doi.org/10.1007/978-3-319-12060-7_13` (cit. on p. 64).

[Car+14]   Claude Carlet et al. "Achieving side-channel high-order correlation immunity with leakage squeezing". In: *Journal of Cryptographic Engineering* 4.2 (2014), pp. 107–121 (cit. on pp. 29, 62).

[CDP16a]   Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. "Enhancing Dimensionality Reduction Methods for Side-Channel Attacks". In: *Smart Card Research and Advanced Applications: 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*. Ed. by Naofumi Homma and Marcel Medwed. Cham: Springer International Publishing, 2016, pp. 15–33. ISBN: 978-3-319-31271-2. DOI: `10.1007/978-3-319-31271-2_2`. URL: `http://dx.doi.org/10.1007/978-3-319-31271-2_2` (cit. on pp. 16, 43).

[CDP16b]   Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. "Kernel Discriminant Analysis for Information Extraction in the Presence of Masking". In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2016, pp. 1–22 (cit. on pp. 16, 61).

[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. 43).

[Cha+99]   Suresh Chari et al. "Towards sound approaches to counteract power-analysis attacks". In: *Annual International Cryptology Conference*. Springer. 1999, pp. 398–412 (cit. on pp. 21, 28, 62, 63).

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

com/science/article/pii/S0031320399001399 (cit. on pp. 45, 53).

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. "An efficient method for random delay generation in embedded software". In: *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 156–170 (cit. on p. 28).

[CK10] Jean-Sébastien Coron and Ilya Kizhvatov. "Analysis and improvement of the random delay countermeasure of CHES 2009". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2010, pp. 95–109 (cit. on p. 28).

[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. 43).

[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. 24, 43, 44, 48).

[CK18] Marios O Choudary and Markus G Kuhn. "Efficient, Portable Template Attacks". In: *IEEE Transactions on Information Forensics and Security* 13.2 (2018), pp. 490–501 (cit. on pp. 44, 59).

[CL06] Tonatiuh Peña Centeno and Neil D Lawrence. "Optimising kernel parameters and regularisation coefficients for non-linear discriminant analysis". In: *The Journal of Machine Learning Research* 7 (2006), pp. 455–491 (cit. on p. 69).

[Cor+13] Jean-Sébastien Coron et al. "Higher-order side channel security and mask refreshing". In: *International Workshop on Fast Software Encryption*. Springer. 2013, pp. 410–424 (cit. on p. 28).

[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. 23, 24, 25).

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297 (cit. on p. 39).

[DC+15] Thomas De Cnudde et al. "Higher-order threshold implementation of the AES S-box". In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2015, pp. 259–272 (cit. on p. 28).

[DH76] Whitfield Diffie and Martin Hellman. "New directions in cryptography". In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654 (cit. on p. 6).

[DS15] François Durvaux and François-Xavier Standaert. "From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces". In: *IACR Cryptology ePrint Archive* (2015), p. 536 (cit. on p. 64).

[Dur+15] François Durvaux et al. "Efficient selection of time samples for higher-order DPA with projection pursuits". In: *Constructive Side-Channel Analysis and Secure Design*. Springer, 2015, pp. 34–50 (cit. on pp. 44, 63).

[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. 43).

[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. 44).

[Fuk90]   Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition (2Nd Ed.)* San Diego, CA, USA: Academic Press Professional, Inc., 1990. ISBN: 0-12-269851-7 (cit. on pp. 51, 53).

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

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

[Gie+10]   Benedikt Gierlichs et al. "Revisiting higher-order DPA attacks". In: *Cryptographers? Track at the RSA Conference*. Springer. 2010, pp. 221–234 (cit. on p. 29).

[GLRP06]   Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. "Templates vs. stochastic methods". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2006, pp. 15–29 (cit. on pp. 25, 26).

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

[GP99]   Louis Goubin and Jacques Patarin. "DES and differential power analysis the ?Duplication? method". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 1999, pp. 158–172 (cit. on p. 28).

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

[GR17]   Dahmun Goudarzi and Matthieu Rivain. "How Fast Can Higher-Order Masking Be in Software?" In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2017, pp. 567–597 (cit. on p. 28).

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

[Hos+11]   Gabriel Hospodar et al. "Machine learning in side-channel analysis: a first study". English. In: *Journal of Cryptographic Engineering* 1.4 (2011), pp. 293–302. ISSN: 2190-8508. DOI: 10.1007/s13389-011-0023-x. URL: http://dx.doi.org/10.1007/s13389-011-0023-x (cit. on p. 39).

[Hua+02] Rui Huang et al. "Solving the small sample size problem of LDA". In: *Pattern Recognition, 2002. Proceedings. 16th International Conference on*. Vol. 3. 2002, 29–32 vol.3. DOI: `10.1109/ICPR.2002.1047787` (cit. on pp. 45, 54).

[HZ12] Annelie Heuser and Michael Zohner. "Intelligent Machine Homicide". English. In: *Constructive Side-Channel Analysis and Secure Design*. Ed. by Werner Schindler and SorinA. Huss. Vol. 7275. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 249–264. ISBN: 978-3-642-29911-7. DOI: `10.1007/978-3-642-29912-4_18`. URL: `http://dx.doi.org/10.1007/978-3-642-29912-4_18` (cit. on p. 39).

[IPS02] James Irwin, Dan Page, and Nigel P Smart. "Instruction stream mutation for non-deterministic processors". In: *Application-Specific Systems, Architectures and Processors, 2002. Proceedings. The IEEE International Conference on*. IEEE. 2002, pp. 286–295 (cit. on p. 28).

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. "Private circuits: Securing hardware against probing attacks". In: *Annual International Cryptology Conference*. Springer. 2003, pp. 463–481 (cit. on pp. 20, 28).

[JPS05] Marc Joye, Pascal Paillier, and Berry Schoenmakers. "On second-order differential power analysis". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2005, pp. 293–308 (cit. on p. 29).

[JS17] Anthony Journault and François-Xavier Standaert. "Very high order masking: Efficient implementation and security evaluation". In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2017, pp. 623–643 (cit. on p. 28).

[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. 43).

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

[Koc94] Cetin Kaya Koc. "High-Speed RSA Implementation Version 2.0". In: *RSA Security* (1994) (cit. on p. 19).

[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. 8).

[LBM14] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. "Power analysis attack: an approach based on machine learning". In: *International Journal of Applied Cryptography* 3.2 (2014), pp. 97–115 (cit. on p. 39).

[LBM15] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. "A machine learning approach against a masked AES". In: *Journal of Cryptographic Engineering* 5.2 (2015), pp. 123–139 (cit. on p. 39).

[LCY92] Ke Liu, Yong-Qing Cheng, and Jing-Yu Yang. "A generalized optimal set of discriminant vectors". In: *Pattern Recognition* 25.7 (1992), pp. 731–739 (cit. on p. 53).

[Ler+15]   Liran Lerman et al. "Template Attacks vs. Machine Learning Revisited (and the Curse of Dimensionality in Side-Channel Analysis)". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2015, pp. 20–33 (cit. on p. 39).

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

[Lio+14]   Rokach Lior et al. *Data mining with decision trees: theory and applications*. Vol. 81. World scientific, 2014 (cit. on p. 39).

[Lom+14]   Victor Lomné et al. "How to Estimate the Success Rate of Higher-Order Side-Channel Attacks". English. In: *Cryptographic Hardware and Embedded Systems CHES 2014*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 35–54. ISBN: 978-3-662-44708-6. DOI: 10 . 1007 / 978 − 3 − 662 − 44709−3_3. URL: http://dx.doi.org/10.1007/978−3−662−44709−3_3 (cit. on p. 63).

[LPR13]   Victor Lomné, Emmanuel Prouff, and Thomas Roche. "Behind the scene of side channel attacks". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2013, pp. 506–525 (cit. on p. 25).

[LRP07]   Kerstin Lemke-Rust and Christof Paar. *Gaussian mixture models for higher-order side channel analysis*. Springer, 2007 (cit. on p. 63).

[LZO06]   Tao Li, Shenghuo Zhu, and Mitsunori Ogihara. "Using discriminant analysis for multi-class classification: an experimental investigation". In: *Knowledge and Information Systems* 10.4 (2006), pp. 453–472. ISSN: 0219-3116. DOI: 10.1007/s10115−006−0013−y. URL: http://dx.doi.org/10.1007/s10115−006−0013−y (cit. on p. 69).

[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. 22).

[Mas94]   James L Massey. "Guessing and entropy". In: *Information Theory, 1994. Proceedings., 1994 IEEE International Symposium on*. IEEE. 1994, p. 204 (cit. on p. 27).

[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. 44, 48, 49).

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

[Mes00a]   Thomas S Messerges. "Securing the AES finalists against power analysis attacks". In: *International Workshop on Fast Software Encryption*. Springer. 2000, pp. 150–164 (cit. on p. 28).

[Mes00b]   Thomas S Messerges. "Using second-order power analysis to attack DPA resistant software". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2000, pp. 238–251 (cit. on p. 29).

[MHM13]   Zdenek Martinasek, Jan Hajny, and Lukas Malina. "Optimization of power analysis using neural network". In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2013, pp. 94–107 (cit. on p. 39).

[MMS01]   David May, Henk L Muller, and Nigel P Smart. "Non-deterministic processors". In: *Australasian Conference on Information Security and Privacy*. Springer. 2001, pp. 115–129 (cit. on p. 28).

[MMT15]   Zdenek Martinasek, Lukas Malina, and Krisztina Trasy. "Profiling power analysis attack based on multi-layer perceptron network". In: *Computational Problems in Science and Engineering*. Springer, 2015, pp. 317–339 (cit. on p. 39).

[Moo+02]   Simon Moore et al. "Improving smart card security using self-timed circuits". In: *Asynchronous Circuits and Systems, 2002. Proceedings. Eighth International Symposium on*. IEEE. 2002, pp. 211–218 (cit. on p. 28).

[Moo+03]   Simon Moore et al. "Balanced self-checking asynchronous logic for smart card applications". In: *Microprocessors and Microsystems* 27.9 (2003), pp. 421–430 (cit. on p. 28).

[MOP08]   Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*. Vol. 31. Springer Science & Business Media, 2008 (cit. on pp. 21, 25).

[Mor+11]   Amir Moradi et al. "Pushing the limits: a very compact and a threshold implementation of AES". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2011, pp. 69–88 (cit. on p. 28).

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

[MPP16]   Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. "Breaking Cryptographic Implementations Using Deep Learning Techniques". In: *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer. 2016, pp. 3–26 (cit. on p. 39).

[MZ13]   Zdenek Martinasek and Vaclav Zeman. "Innovative method of the power analysis". In: *Radioengineering* 22.2 (2013), pp. 586–594 (cit. on p. 39).

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

[OC14]   Colin O'Flynn and Zhizhang David Chen. "ChipWhisperer: An open-source platform for hardware embedded security research". In: *Constructive Side-Channel Analysis and Secure Design*. Springer, 2014, pp. 243–260 (cit. on pp. 55, 68).

[Osw+05]   Elisabeth Oswald et al. "A side-channel analysis resistant description of the AES S-box". In: *International Workshop on Fast Software Encryption*. Springer. 2005, pp. 413–423 (cit. on p. 28).

[Osw+06]   Elisabeth Oswald et al. "Practical second-order DPA attacks for masked smart card implementations of block ciphers". In: *Cryptographers? Track at the RSA Conference*. Springer. 2006, pp. 192–207 (cit. on p. 29).

[Ou+17]   Changhai Ou et al. *Manifold Learning Towards Masking Implementations: A First Study*. Cryptology ePrint Archive, Report 2017/1112. `https://eprint.iacr.org/2017/1112`. 2017 (cit. on p. 61).

[OWW13]   Yossef Oren, Ofir Weisse, and Avishai Wool. "Practical Template-algebraic Side Channel Attacks with Extremely Low Data Complexity". In: *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*. HASP '13. Tel-Aviv, Israel: ACM, 2013, 7:1–7:8. ISBN: 978-1-4503-2118-1. DOI: `10.1145/2487726.2487733`. URL: `http://doi.acm.org/10.1145/2487726.2487733` (cit. on p. 27).

[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. 26).

[Par]   TELECOM ParisTech. "DPA Contest 4". In: (). http://www.DPAcontest.org/v4/. URL: `http://www.DPAcontest.org/v4/` (cit. on p. 48).

[Pee+05]   Eric Peeters et al. "Improved higher-order side-channel attacks with FPGA experiments". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2005, pp. 309–323 (cit. on p. 29).

[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. 24).

[PM05]   Thomas Popp and Stefan Mangard. "Masked dual-rail pre-charge logic: DPA-resistance without routing constraints". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2005, pp. 172–186 (cit. on p. 28).

[PR13]   Emmanuel Prouff and Matthieu Rivain. "Masking against side-channel attacks: A formal security proof". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2013, pp. 142–159 (cit. on p. 28).

[PRB09]   Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. "Statistical Analysis of Second Order Differential Power Analysis". In: *IEEE Trans. Computers* 58.6 (2009), pp. 799–811. DOI: `10.1109/TC.2009.15`. URL: `http://doi.ieeecomputersociety.org/10.1109/TC.2009.15` (cit. on p. 29).

[PS08]   Gilles Piret and F-X Standaert. "Security analysis of higher-order Boolean masking schemes for block ciphers (with conditions of perfect masking)". In: *IET Information Security* 2.1 (2008), pp. 1–11 (cit. on p. 29).

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

[RGV12]   Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. "Selecting Time Samples for Multivariate DPA Attacks". English. In: *Cryptographic Hardware and Embedded Systems CHES 2012*. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 155–174. ISBN: 978-3-642-33026-1. DOI: 10.1007/978-3-642-33027-8_10. URL: http://dx.doi.org/10.1007/978-3-642-33027-8_10 (cit. on p. 63).

[Riv91]   Ronald L Rivest. "Cryptography and machine learning". In: *International Conference on the Theory and Application of Cryptology*. Springer. 1991, pp. 427–439 (cit. on p. 39).

[RO05]   Christian Rechberger and Elisabeth Oswald. "Practical Template Attacks". English. In: *Information Security Applications*. Ed. by ChaeHoon Lim and Moti Yung. Vol. 3325. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 440–456. ISBN: 978-3-540-24015-0. DOI: 10.1007/978-3-540-31815-6_35. URL: http://dx.doi.org/10.1007/978-3-540-31815-6_35 (cit. on p. 25).

[RP10]   Matthieu Rivain and Emmanuel Prouff. "Provably secure higher-order masking of AES". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2010, pp. 413–427 (cit. on p. 28).

[RS09]   Mathieu Renauld and François-Xavier Standaert. "Algebraic Side-Channel Attacks." In: *Inscrypt* 6151 (2009), pp. 393–410 (cit. on p. 25).

[RSA78]   Ronald L Rivest, Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126 (cit. on p. 6).

[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. 25).

[SA08]   François-Xavier Standaert and Cedric Archambeau. "Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages". English. In: *Cryptographic Hardware and Embedded Systems CHES 2008*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 411–425. ISBN: 978-3-540-85052-6. DOI: 10.1007/978-3-540-85053-3_26. URL: http://dx.doi.org/10.1007/978-3-540-85053-3_26 (cit. on pp. 43, 51, 53).

[SM99]   Bernhard Schölkopf and Klaus-Robert Mullert. "Fisher discriminant analysis with kernels". In: *Neural networks for signal processing IX* 1 (1999), p. 1 (cit. on pp. 68, 69).

[SMY09]   François-Xavier Standaert, TalG. Malkin, and Moti Yung. "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks". English. In: *Advances in Cryptology - EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 443–461. ISBN: 978-3-642-01000-2. DOI: 10.1007/978-3-642-01001-9_26. URL: http://dx.doi.org/10.1007/978-3-642-01001-9_26 (cit. on p. 27).

[SP06]   Kai Schramm and Christof Paar. "Higher order masking of the AES". In: *Cryptographers? Track at the RSA Conference*. Springer. 2006, pp. 208–225 (cit. on p. 28).

[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. 44, 48).

[SPQ05]   F-X Standaert, Eric Peeters, and J-J Quisquater. "On the masking countermeasure and higher-order power analysis attacks". In: *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*. Vol. 1. IEEE. 2005, pp. 562–567 (cit. on p. 29).

[SSM98]   Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear component analysis as a kernel eigenvalue problem". In: *Neural computation* 10.5 (1998), pp. 1299–1319 (cit. on pp. 66, 83).

[TM97]   Mitchell T. M. *Machine Learning*. McGraw-Hill, New York, 1997 (cit. on p. 31).

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

[VC+12]   Nicolas Veyrat-Charvillon et al. "Shuffling against side-channel attacks: A comprehensive study with cautionary note". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2012, pp. 740–757 (cit. on p. 28).

[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. 26).

[WM97]   David H Wolpert and William G Macready. "No free lunch theorems for optimization". In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82 (cit. on p. 38).

[WW04]   Jason Waddle and David Wagner. "Towards Efficient Second-Order Power Analysis". English. In: *Cryptographic Hardware and Embedded Systems - CHES 2004*. Ed. by Marc Joye and Jean-Jacques Quisquater. Vol. 3156. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 1–15. ISBN: 978-3-540-22666-6. DOI: 10.1007/978-3-540-28632-5_1. URL: http://dx.doi.org/10.1007/978-3-540-28632-5_1 (cit. on p. 29).

[WW98]   Jason Weston and Chris Watkins. *Multi-class support vector machines*. Tech. rep. Citeseer, 1998 (cit. on p. 39).

[YY01]   Hua Yu and Jie Yang. "A direct LDA algorithm for high-dimensional data with application to face recognition". In: *Pattern Recognition* 34 (2001), pp. 2067–2070 (cit. on pp. 45, 54).

[Zho+17] Xinping Zhou et al. "A Novel Use of Kernel Discriminant Analysis as a Higher-Order Side-Channel Distinguisher". In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2017, pp. 70–87 (cit. on p. 61).