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

August 13, 2018

Contents

List of Figures

List of Tables

List of Abbreviations

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

List of Symbols

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

Part I

Context and State of the Art

Chapter 1

Context

1.1 Introduction to Cryptography

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

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

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

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

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

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

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

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

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

1.1.1 Description of AES

The *Advanced Encryption Standard* (AES) has been standardized in 2001 by the United States governmental agency *National Institute of Standards and Technology* (NIST), through the *Federal Information Processing Standards Publication 197* (FIPS PUB 197) [NIS]. It is a *block cipher*, meaning that the encryption and decryption of the AES are functions that take as input a string (respectively the plaintext or the ciphertext) of fixed length over the binary alphabet. Indeed, the AES operates on blocks of 128 bits.[1] There exist three versions of AES, characterized by the size of the used key: 128, 192 or 256 bits. The encryption is done by rounds. The number of executed

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

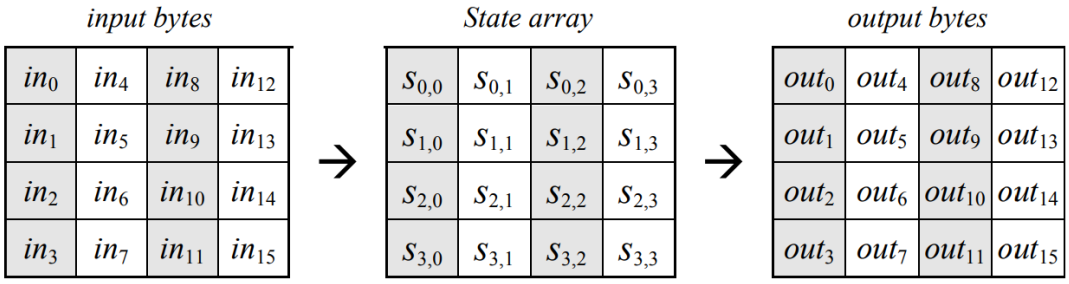


FIGURE 1.1: State array input and output. Source: [NIS].

rounds depends on the key size (10 rounds for 128 bits, 12 for 192 and 14 pour 256). The basic processing unit in AES algorithm is a byte. For AES internal operations, bytes are arranged on a two-dimensional array called the *state*, denoted s. Such a state has 4 rows and 4 columns, thus contains 16 bytes. The byte lying at the i-th row, j-th column of s will be denoted by $s_{i,j}$ for $i, j \in \{0, 1, 2, 3\}$. The 16 input bytes and the 16 output bytes are indexed column-wise as shown in Fig. 1.1. Each element $s_{i,j}$ of a state is mathematically seen as an element of the *Rjindael finite field*, defined as $GF(2^8) = \mathbb{Z}/2\mathbb{Z}[X]/P(X)$ where $P(X) = X^8 + X^4 + X^3 + X + 1$. Five functions are performed during the AES, named KeySchedule, AddRoundKey, SubBytes, ShiftRows and MixColumns. At high level the AES algorithm is described hereafter:

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

Round 0:

AddRoundKey

Rounds 1 to penultimate:

SubBytes

ShiftRows

MixColumns

AddRoundKey

Last Round:

SubBytes

ShiftRows

AddRoundKey

A description of the five functions is provided hereafter.

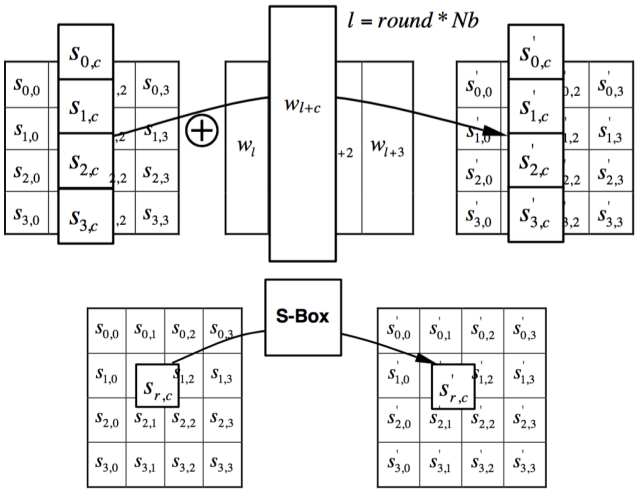


FIGURE 1.2: AddRoundKey (top) and SubBytes (bottom) operate over the State byte by byte, independently. Source: [NIS].

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:

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

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

ShiftRows

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

MixColumns

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

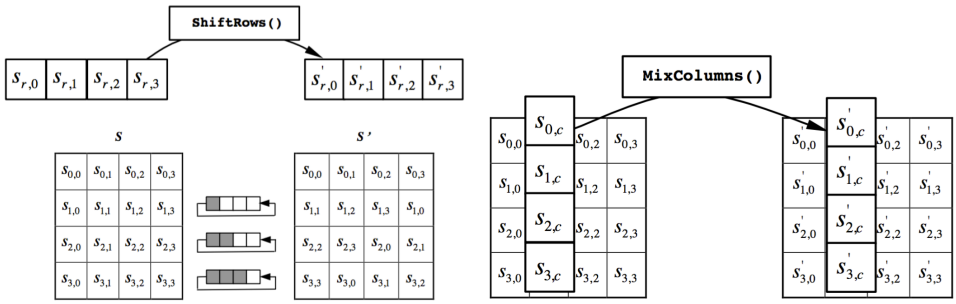


FIGURE 1.3: ShiftRows operates over the State rows. MixColumns operates over the State columns. Source: [NIS].

KeySchedule

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

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

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

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

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

1.2 Secure Components

As we have seen in the previous section, modern cryptography proposes solutions to secure communications that ask for electronic computations and repose their security over some secret keys. Keys are represented as long bit strings, very hard to be memorized by users. Thus, keys need to be stored in a secure medium, and never delivered in clear over insecure channels. Smart cards were historically conceived as a practical solution to such a key storage issue: they consist in small devices a user can easily carry around with, which not only store secret keys, but also are able to internally perform cryptographic operations, in such a way that keys are never

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

delivered. The registrations of a first patent describing memory cards by Roland Moreno in 1974 [Mor74], and of a second one describing cards equipped with microcontrollers by Michel Ugon in 1977 [Ugo77] are often referred to date the smart card invention, finally produced for the first time in 1979. Smart cards are pocket-sized plastic-made cards equipped with a secure component, which is typically an integrated circuit containing some computational units and some memories.

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

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

1.2.1 Embedded Cryptography Vulnerabilities

1.2.1.1 Side-Channel Attacks

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

TABLE 1.1: Classification of Harware Attacks

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

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

1.2.1.2 A Classification of the Attacks against Secure Components

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

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

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

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

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

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

In the literature, the term Side-Channel Attacks (SCAs)[3] commonly refers to the passive non-invasive attacks. Nevertheless, the techniques proposed under the name of SCAs, that always require the acquisition of some signals, might also include attacks where the device is unpacked, in order to improve the signal amplitude. In this sense, SCAs belong to the semi-invasive group of attacks as well. Similarly, active non-invasive attacks are often referred to as *Fault Injection Attacks*, that might also be run in a semi-invasive way.

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

1.2.2 Certification of a Secure Hardware - The Common Criteria

In previous paragraphs we have evoked the great diffusion of the cryptographic devices and the existence of a wide range of attacks exploiting vulnerabilities coming from the way cryptography is embedded. These two factors imply a great risk related to the production and commercialization of such devices, and justify the importance and necessity to ensure reliability on their security claims. This necessity lead to the arise of several guidelines and standards for their evaluation. The international standard ISO/IEC 15408, also known as *Common Criteria for Information*

[3]Commonly the acronym SCA stands for "Side-Channel Analysis". Nevertheless, in this thesis it will stand for "Side-Channel Attacks".

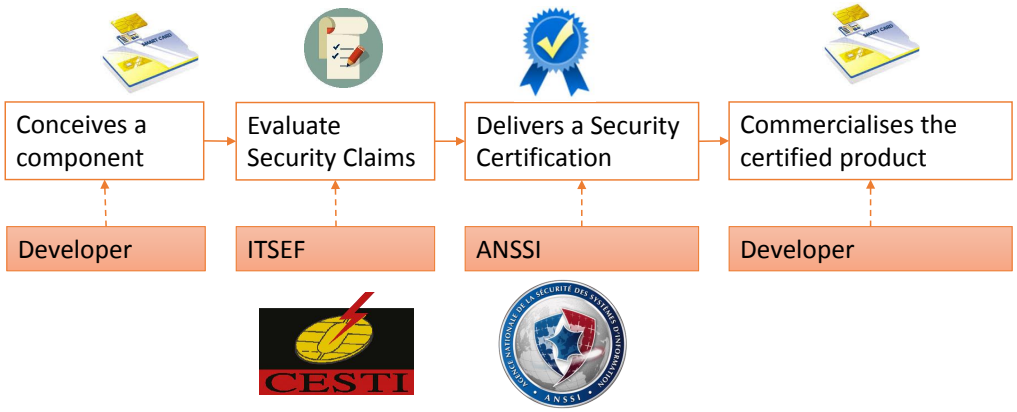


FIGURE 1.4: The actors of French Certification Scheme

Technology Security Evaluation (abbreviated as *Common Criteria* or simply *CC*) represents one of the strongest efforts in standardization, unifying in 1999 three previously existing standards:

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

1.2.2.1 The actors

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

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

TABLE 1.2: Evaluation Assurance Levels

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

1.2.2.2 The Target of Evaluation and the security objectives

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

1.2.2.3 Evaluation Assurance Level and Security Assurance Requirements

In CC seven *Evaluation Assurance Level* (EAL) are defined. They determine the quantity and complexity of the tasks the evaluator has to effectuate, thus specifying the insurance strength. The EAL are defined in insurance increasing order, so that the EAL1 has the lowest verification exigences while EAL7 has the highest ones. In Table 1.2 the objectives given by the CC for each EAL are resumed.

During the process of evaluation, the SFRs of the TOE have to be verified according to the claimed EAL. To this end, the evaluation is divided into six classes of *Security Assurance Requirement* (SAR). Five of this classes are the so-called *conformity* classes, and one is the *vulnerability assessment* class. Each class is sub-divided in several *families* (excepted the vulnerability assessment class, which only contains one family), and the evaluators are charged to check each requirement corresponding to these families. The Table 1.3 resumes the SAR classes and their families. For each family a grade is assigned following precise specifications detailed in CC, and

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

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

1.2.2.4 The AVA_VAN family and the Attack Potential

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

In the case of smart cards, the evaluation systematically includes the AVA_VAN5 grade, thus the testing plan is asked to be as complete as possible. The state-of-the-art of the attacks is periodically upgraded by the *JIL*[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 successful attack tested by the evaluators must

[4]Joint Interpretation Library

have been rated at least 31.

1.2.2.5 The Evaluation Technical Report

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

1.3 This thesis objectives and contributions

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

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

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

1.3.1 Foreword of this Thesis: Research of Points of Interest

To perform a Side-Channel Attack, the monitoring of unintentional channels leaking from the attacked device is usually performed through an oscilloscope that samples continuous analog signals and turns them into discrete digitalized sequences. Such sequences are often referred to as *traces*. To allow a deep inspection of the device, the sampling rate of the oscilloscope needs to be high, leading very often to a high dimensionality of such traces. Nevertheless, it is expected that only a limited number of time samples are relevant for a SCA: those that are statistically dependent on the sensitive variable that is exploited to run the attack. Such time samples are called *Points of Interest* (PoIs). In the literature a few of different statistics was proposed and exploited to select such PoIs in a preliminary attack phase, in order to reduce both time and memory complexity of the attacks. A brief overview of such statistics is proposed in Sec. 2.5.2. The foreword of this thesis was to propose new methods to research and characterise the PoIs, in order to ameliorate and possibly optimise the preliminary attack phase consisting in their selection.

1.3.2 Dimensionality Reduction Approach

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

Nowadays every device needing to obtain an AVA_VAN5 grade is equipped of specific countermeasures against SCAs. A brief overview of some classic and public principles providing efficient countermeasure is provided in Sec. **??**. Among them, the *masking*, or *sharing*, countermeasures may be considered the most effective ones. Beyond the formal proofs of their efficiency provided in the literature [ISW03; PR13; Bar+15], they are the ones that most likely require a strong adaptation of the attack strategy in order to be defeated. Indeed, when an effective masking scheme is implemented, each sensitive variable of the original computation is split into shares randomly drawn, in such a way that any proper subset of shares is statistically independent of the sensitive variable itself. Computation of cryptographic primitives

is done accessing only the random shares, with intermediate steps computing only the shares of the result. This forces the attacker to work with the joint distributions of the signal at the time samples where the shares are being accessed. In other words, point-wise statistics to retrieve PoIs are completely inefficient in presence of a masking countermeasure, since each time sample is by itself statistically independent from any sensitive variable. Moreover, interesting joint distributions have to be studied at their higher-order statistical moments to retrieve sensitive-data dependencies, implying that any linear method to combine time samples is inefficient as well. To sum up, the issue of selecting PoIs or applying dimensionality reduction to side-channel traces protected by masking presents challenging difficulties. Such a hardness is mitigated when the attacker is able to perform a profiling phase during which he has the knowledge of the random values assigned to the shares during execution. In practice it is not always the case, so in this thesis we tackle the issue when such knowledge is absent, and we propose in Chapter 5, on the basis of a work presented at CARDIS 2016 [CDP16b], to deploy Kernel Fisher Discriminant Analysis (KDA) as a solution. This is an extension of the LDA dimensionality reduction technique, allowing applying some strategy to non-linearly combine time samples.

1.3.3 Towards Machine Learning and Neural Networks Approach

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

with masked implementations.

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

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

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

danger in embedded security domain, leading to powerful Side-Channel Attacks.

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

Chapter 2

Introduction

2.1 Notations and Probability and Statistics Recalls

Basic Notations. In this thesis we use calligraphic letters as $\mathcal{X}$ to denote sets, the corresponding upper-case letter X to denote random variables (random vectors $\vec{X}$ if with an arrow) over $\mathcal{X}$, and the corresponding lower-case letter x (resp. $\vec{x}$ for vectors) to denote realizations of X (resp. $\vec{X}$). The cardinality of a set $\mathcal{X}$ is denoted by $|\mathcal{X}|$. Matrices will be denoted with bold capital letters. When the vectors' orientation minds, they are understood as column vectors. The i-th entry of a vector $\vec{x}$ is denoted by $\vec{x}[i]$, while the transposed of a vector $\vec{x}$ is denoted as $\vec{x}^\intercal$. We will use the transposed mark to refer to row vectors $\vec{x}^\intercal$. In general the ith observation of a random vector X will be denoted by x_i. Observations will sometimes be grouped into datasets $\mathcal{D} = \{x_1, \ldots, x_N\}$. Throughout this thesis, a finite set $\mathcal{Z} = \{s_1, \ldots, s_{|\mathcal{Z}|}\}$ will be often considered: it will always denote the possible values for a *sensitive variable* Z (see later). Its elements are sometimes encoded via a so-called *one-hot-encoding*: to each element s_j a $|\mathcal{Z}|$-dimensional vector $\vec{s_j}$ is associated, with all entries equal to 0 and the j-th entry equal to 1: $s_j \to \vec{s_j} = (0, \ldots, 0, \underbrace{1}_{j}, 0, \ldots, 0)$. We will denote by s a generic element of $\mathcal{Z}$, when useless to specify its index s_i for the context.

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

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

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

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

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

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

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

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

by the following expressions, in unidimensional and multidimensional case:

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

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

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

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

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

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

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

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

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

2.2 Introduction to Side-Channel Attacks

Side-Channel Attacks (SCA) belong to the cryptanalysis domain, since they aim to breach cryptographic security systems. Usually their goal is to retrieve a secret variable of a cryptographic algorithm, typically a secret key. They distinguish from classic mathematical cryptanalysis techniques by the fact that they are based on information gained from the physical implementation of a cryptosystem, rather than theoretical weaknesses in the algorithms. The possibility to physically observe the electronic device that performs the cryptographic computations, allows Side-Channel Attacks to go beyond the cryptographic complexity that assures resistance against classical cryptanalysis strategies. Indeed, no matter the size of the secret variables

manipulated by the algorithm and the algebraic complexity of the encrypting/decrypting operations, an physical implementation of any algorithm always handles variables of a bounded size, that depends on the hardware architecture of the cryptographic device. For example, in an 8-bit architecture an AES with 128-bit-sized key will be necessarily implemented as multiple partial computations over 8-bit blocks of data. In classical cryptanalysis, the typical attacker model faces to a black-box that performs the cryptographic algorithm: an attacker may make queries to the black-box, asking for ciphertexts of given plaintexts or viceversa. The black-box acts as a function that output the asked value, but does not provide any information about partial computations. On the contrary, a side-channel attacker is said to face to a *grey-box* model: he has a way to obtain noised information about partial computations. This allows him to perform a *divide-and-conquer* strategy: if his goal is to retrieve the full 128-bit AES key, he will smartly divide his problem into the recovery of small parts of such keys at time, called *key chunks*,[1] making the complexity of the attack significantly drop.

2.2.1 An Overview

Since the seminal paper by Paul Kocher in 1996 [Koc96], side-channel analysis domain has developed fast, together with its flourish literature. Without being exhaustive, the last literature includes: proposals for new kind of exploitable signals, proposals for useful statistical tools, new attacks strategies and routines, analysis of side-channel vulnerabilities of well-specified cryptographic algorithms, side-channel countermeasures, formal proofs of countermeasures' security claims, discussions about tools and metrics to compare side-channel attacks and strategies, reports of real-case successful attacks, and a few attempts to unify the side-channel literature under some comprehensive frameworks. The contributions we present in this thesis may be resumed as proposals for statistical tools for some specific attack contexts. The aim of the following part of this section is not to provide a comprehensive state-of-the-art of side-channel domain, but to provide the reader with the necessary concepts to understand such contributions, and get a view of the contexts in which they can provide improvements to the state-of-the-art. To this aim we propose a brief overview of the main properties that define and characterise a side-channel attack among others. To describe a side-channel attacks, we identified are the following characteristics:

- the physical nature of the exploited signals,
- the chosen sensitive variables,
- the strategy family,

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

- the *shape* of the attack,
- and the attacker knowledge.

In the following sections we will briefly describe these points, dwelling on aspects that mainly concern our contributions, *e.g.* the *advanced attack* strategy and the concept of *profiing attack*.

2.2.2 Physical Nature of the Exploited Signals

As already introduced in Sec. 1.2.1.1, a SCA may exploit signals obtained by the observation of different kind of *side channels*. Mainly exploited physical quantities are the power consumption, the electromagnetic emanation, the elapsing time and the acoustic emanation.

2.2.3 Sensitive Variables

Physical signals are acquired *via* appropriate instrumentation, and collected into vectors called *traces* (or *acquisitions*). They will be denoted by $\vec{x}_i$ and considered as observations of a random real vector $\vec{X}$, where each coordinate corresponds to a time sample of the acquired signal. They are then interpreted as noisy observations of the intermediate variables handled by the device during the execution. An attacker is in particular interested to the so-called *sensitive variables*: they are quantities handled during the processing that depend somehow on a secret of the implementation, and not only on public variables, as a plaintext or an algorithm constant. Side-channel analysis acts clearing traces from noise, in such a way to determine with the highest possible precision the association between a trace (or a set of traces) and the value taken by the target *sensitive variable* Z during its (their) acquisition. For an attack, a single or several sensitive variables may be targeted, and its/their algebraic relation with the secret key serves to complete the attack. Actually, *sensitive variables* would be more appropriately called *sensitive targets*, since they might not be variable. Some typical examples of sensitive variables include:

- $Z = K$ with K a secret subkey - this is the most direct choice for a sensitive target, nevertheless it is often not variable, since in some cases a device always manipulates the same key for a given embedded primitive. When the target is not variable we are performing a *simple attack* (see Sec. 2.2.4.1);
- a cryptographic variable that depends on a sufficiently small subkey and a part of a known input variable E: $Z = f(K, E)$ - this is the most classical choice to perform a so-called *differential* or *advanced* SCA (see 2.4);

- any function of a cryptographic variable (ex: $\mathrm{HW}(f(K, E))$), where $\mathrm{HW}(\cdot)$ represents the Hamming weight operation, *i.e.* the operation that counts the number of 1's in the binary string representing the entry. Sometimes, as for example we will see in Chapter 5 (see Sec. 5.4.3) it can be interesting not to target a variable but a non-injective function of a variable as its Hamming weight; when the identity function is applied we are in the previous case;
- an operation (ex: $Z \in \{\mathrm{square}, \mathrm{multiply}\}$)
- a register (ex: Z is the register used to store results of intermediate operations in a Montgomery ladder implementation of RSA)

In this thesis we will try as much as possible to abstract from the form of the sensitive variable, thinking of any entity Z that assumes values in a finite set $\mathcal{Z}$ and whose value permits an attacker to make inference on a secret of the implemented algorithm.

2.2.4 The Strategy Family

The wide range of attack strategies, together with their still-evolving taxonomy, makes the task of group attack strategies very hard. We propose here a simplified grouping into three strategy families:

- the *Simple Attacks*,
- the *Collision Attacks*,
- the *Advanced Attacks*.

We highlight the fact that such a grouping is not sharp and impermeable in literature.

2.2.4.1 Simple Attacks

In simple attacks, the relevant information is obtained directly from the observation of trace pattern, without necessarily apply statistical tools and often at the naked eye. Such a direct analysis is sometimes referred to as *Simple Power Analysis* (SPA). The sensitive variable coincides in general with the secret key (or a chunk of it). Typical targets for SPA attacks are cryptographic devices in which some operations requires variable timing instructions, or in which the execution path depends on the key. For example, considering software-implementation, branching to different instructions may occur when a secret key chunk has a specific value, in general dealing with operations as comparisons, multiplications or exponentiations. A typical example of leaks allowing simple attack is depicted in Fig. 2.1: the depicted trace shows a sequence of squares and multiplications performed by a device computing

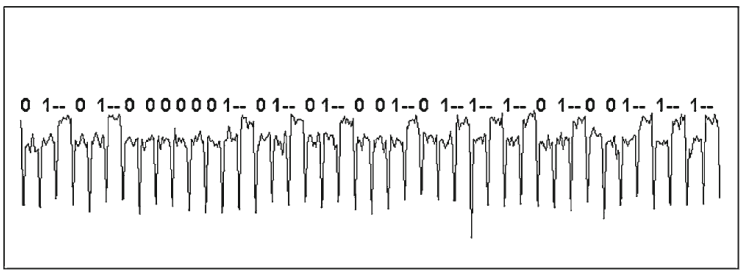


FIGURE 2.1: Simple attack against RSA implementation. Source: [Koc+11].

modular exponentiation to implement the RSA algorithm. Multiplications consume more than squares and patterns are recognizable to the eye. The sequence of patterns directly reveals the secret key. A characteristic of simple attacks is that they do not require in general to observe the variation of the side-channel signals under the variation of the algorithm entries, thus they are sometimes referred to as *one-trace attacks*, since the observation of a single trace may be sufficient to perform them. In literature the terms *simple attacks* and *one-trace attacks* are sometimes considered equivalent, as *e.g.* in [Cla+12]. Anyway, we aim to include in the *simple attacks* family those attacks for which many observations are acquired with fixed entry parameters and by consequence in which the observed leakage always corresponds to a fixed value of Z. The attacker may exploit the several acquisitions in mainly two ways: he computes their average before performing the attack (as done for example in the simple attack proposed by Mangard in 2002 [Man02], and ameliorate in 2014 by Clavier *et al.* [CMW14]), aiming to reduce the noise influence, or he performs the attack 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 observations allows the attacker to reduce the noise impact, while observing many times the same variable. The relation between the variable and the key chunk being fixed (and being the identity, in general), he will not exploit algebraic relations to ameliorate his inference over the latter. In next chapter, Section 3.1.1, we will describe some classic example of machine learning task. Here we point out the fact a simple attacks exactly correspond to resolving a *classification task* in side-channel context.

2.2.4.2 Collision Attacks

Collision attacks were introduced by Schramm *et al.* in 2003 [SWP03] as a side-channel generalisation of classic cryptanalysis collision attacks, typically used to break cryptographic hash functions. They deduct information about the secret key of a block cipher from the presence or the absence of an internal collision during the encryption (or decryption). A collision has to be intended as the fact that, while processing different inputs, an internal computation acts over the same operand, or outputs the same value. To perform a collision attack, the side-channel attacker is thus not required to interpret side-channel signals to perfectly understand which operation is executed and over which operands. The assumption is weaker: the attacker is supposed to be able to state if two signals (or portions of signals) correspond or not to the same operation. In the seminal work [SWP03], as in several further developments as [LMV04; Sch+04; Bog07; Bog08], sets of several acquisitions under well-chosen entries are exploited to establish, through statistical tools, *e.g.* correlation estimators, whether a collision is present. In the same year 2003, Fouque and Valette [FV03] proposed a collision attack in a context declared by authors more favourable than block ciphers, *i.e.* operations like modular exponentiation.[2] In this context, authors proposed an attack strategy based on the observation and comparison of only two acquisitions. In analogy with simple attacks, often labelled as "one-trace", correlation attacks are thus somehow categorised as "two-traces"attacks, even if this connotation is not always pertinent. In particular, collisions might be searched in different parts of the same trace, *i.e.* in a *horizontal* fashion (see Sec. **??**), leading collision attacks applicable with a single trace, *e.g.* as done in [Cla+10] to attack an RSA implementation protected against simple attacks. However, as we highlighted at the beginning of this section, the terms about side-channel strategy are still not unambiguous in literature, for example, in the summarising work proposed by Kocher *et al.* in 2011 [Koc+11], collision attack are considered as a variant of simple attacks. Again in analogy with simple attacks, and in the same way we observed that simple attacks perfectly declines the machine learning task of classification, we observe that collision attacks are in a strong analogy with another classical machine learning task, *i.e.* the *verification task*, that will be as well introduced in Sec. 3.1.1.

2.2.4.3 Advanced Attacks

In contrast to the *Simple Power Analysis*, enabling simple attacks when large-scale side-channel variations depend on secret values and low noise is present, the so-called *Differential Power Analysis* (DPA) refers to techniques that exploit a statistical

[2] or scalar multiplication in the elliptic curve setting

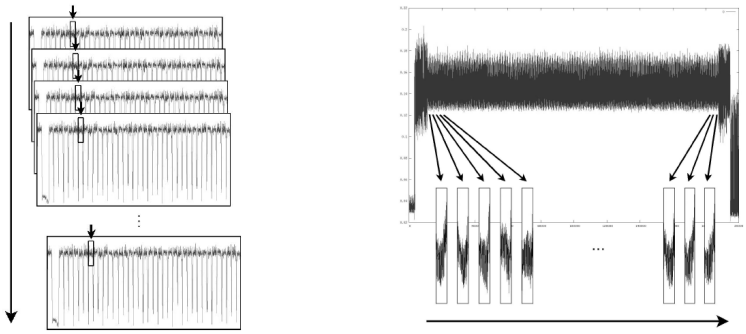


FIGURE 2.2: Vertical (left) and horizontal (right) attack. Source: [Cla+10].

approach to reveal key-dependent lower-scale side-channel variations. DPA techniques enables the so-called *advanced attacks*. With respect to simple attacks, advanced attacks do not require a detailed knowledge of the implementation details, and are able to succeed even dealing with acquisitions containing a considerable amount of noise. The term *Differential* refers to the fact that the approach exploits the small differences in the behaviour of the device while handling varying sensitive variable. By consequence, in contrast to simple attacks, several acquisition have to be observed to perform an advanced attack, under varying values for the chosen sensitive variable. Intuitively, the small differences are get larger by means of averaging over an eventually considerable amount of acquisition. As much noise hides informative differences, as many acquisitions are needed to clear it and make information emerge. Interestingly, the first DPA tool (or *distinguisher*, as will be introduced below) proposed to perform an advanced attack were the so-called *Difference of Means* (DoM) (the method were proposed by [KJJ99], but the name given by [CRR03]), in which differences where exactly looked for by subtraction. A more precise description of the advanced attacks is provided in Sec. 2.4.

2.2.5 The Shape of the Attack

In [Cla+10] a distinction between *vertical* and *horizontal* attacks is proposed, adopted in several posterior publications, *e.g.* [Bau+13] which proposes horizontal approach to attack secure implementations of RSA, or [Bat+16] in which an horizontal approach is used to counteract a *masking* countermeasure (see Sec. **??**). *Vertical attacks*

are intended as techniques analysing the same sample time regions of several side-channel traces, while *horizontal attacks* analyse many portions of a single trace, as depicted in Fig. 2.2. In typical scenarios, horizontal attacks are associated to simple SCAs (see *e.g.* Fig. 2.1), while vertical ones are associated to advanced SCAs. Collision attacks might be vertical or horizontal depending on whether collisions are looked for in a same execution or in more than one execution. Anyway, simple attacks exploiting many acquisitions to reduce the noise impact have a vertical behaviour, and advanced attacks may be performed in an horizontal manner as done *e.g.* in [Bat+16]: this is possible observing that, in the attacked cryptographic algorithm, several different intermediate computations depend on the same sensitive variable, thus the last variable may be observed varying by observing those computations. This kind of approach exploits the algebraic dependency between several intermediate. This concept is at the basis of another special class of SCAs, the so-called *Algebraic Side-Channel Attacks* [RS09; RSVC09; OWW13; OWW14; VGS14]. The Algebraic SCAs combine profiling SCA strategies (see 2.5) to classical cryptanalysis techniques, *i.e.* without necessarily exploiting divide-and-conquer strategy, but retrieving the whole key secret at once solving algebraic systems in which plaintexts, chipertext and potentially all observable intermediate variable are involved. They act in an horizontal manner, and may need more than one acquisition to get a unique key candidate. The nice notion of *rectangular* attacks, introduced in [Bau+13], and denoting attacks that both exploit several portions of a signal, but taking advantage of several acquisitions, probably describes the great majority of the modern attacks.

2.2.6 The Attacker Knowledge

Many aspects of the attacker knowledge on the target implementation may influence his approach. For example the level of knowledge of implementation details may allow or not to perform a simple attack. In an evaluation context, we may assume that an evaluator has open access to implementation details. Nevertheless, we are here interested in distinguish two particular attack scenarios that influence his knowledge about the physical behaviour of the device: the *profiling* and *non-profiling* attacks. As anticipated in Sec. 1.3, when an open sample of the attacked device is available to make a prior characterisation of the leaking signals of a device, we talk about *profiling* attacks. When this is not the case, we talk about *non-profiling* attacks. Profiling attack are the mainly concern of this thesis, and are deeper introduced in Sec. 2.5.

2.3 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 [**unifiedFramework**], a key recovery side-channel attack outputs a vector of key candidates,[3] called *guessing vector* $\vec{g} = [\vec{g}[1], \ldots, \vec{g}[|\mathcal{K}|]]$, in which such candidates are sorted in decreasing order with respect to their likelihood after the attack phase. Being $k^\star$ the right candidate, its *rank* is given by:

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

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

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

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

The guessing entropy [**massey1994guessing**] is defined as the expected rank of the right key:

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

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

2.4 Advanced Attacks

An advanced attack can be summarised in the following five steps:

- acquire side-channel traces $(\vec{x}_i)_{i=1,\ldots,N}$ making entries $(e)_{i=1,\ldots,N}$ vary,
- choose a sensitive variable $f(K, E)$,
- define a *leakage model*, *i.e.* a function $L(Z)$ modelising the side-channel leakage for a given sensitive variable value (some examples are given in Sec. 2.4.1),

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

- for every key chunk hypothesis $k \in \mathcal{K}$ predict the side-channel leakage

$$L_{k,i} = L(f(k, e_i)) \,, \tag{2.9}$$

- statistically compare the hypothetical predictions to the observed side-channel acquisitions, by means of a *distinguisher* Δ (see Sec. 2.4.2):

$$\Delta_k = \Delta((\vec{x}_i)_{i=1,\dots,N}, (L_{k,i})_{i=1,\dots,N}) \,, \tag{2.10}$$

- deduce the key chunk candidate from scores Δ_k, in general coinciding with the key hypothesis that maximises or minimises the scores.

2.4.1 Leakage Models

Classical leakage models are based on the fact that, in CMOS technology (which is used to realise the majority of existing integrated circuits), peaks of power consumption are observable when the output of the gates transition from either a "0"to "1"or a "1"to "0"logic state. For an internal variable Z, examples of classical leakage models $L(Z)$ are:

- *mono-bit model*: the value of one bit of Z,
- *Hamming weight model*: the Hamming weight $\mathrm{HW}(Z)$,
- *Hamming distance model*: the Hamming distance between Z and another intermediate variable Z', defined as $\mathrm{HD}(Z, Z') = \mathrm{HW}(Z \oplus Z')$, supposing *e.g.* that one of the two variable overwrites the other into the same logic states (thus, the number of switches is counted),
- *linear model*: a linear combination of the bits of Z, supposing that some states influence the power consumption more than other,
- *identity model*: the value of Z itself.

2.4.2 Distinguishers

The underlying core hypothesis of an advanced SCA is the following. Given a set of side-channel traces, *i.e.* a set of realizations of a random variable $\vec{X}$, the attacker computes the realizations of a second random variable L_k per each key hypothesis. For the correct key hypothesis, the two random variables $\vec{X}$ and L_k are statistically dependent, while for the wrong key hypothesis they are independent (or at least *apparently* independent, as pointed out in Rem. 2.1). The goal of a distinguisher is to detect the dependencies between the two random variables. Thus, the selected key

candidate is the one that shows an higher dependency value between the predicted leakages and the actual ones.

Remark 2.1. Actually this vision is too strong: a dependency always exists between the wrong key candidate hypothetical leakages and the actual ones (and in literature it has been evidenced under the name of *ghost peaks* in **[brier2004correlation]**), but such a dependency is hard to detect statistically: deterministic functions that link wrong key hypothesis to the correct ones are those that define the cryptographic algorithm, thus in general algebraically complex and highly non-linear. For example, let us consider $Z = \mathrm{Sbox}(K \oplus E)$ as sensitive variable for an AES implementation, and choose in identity leakage model. Let $k^\star$ be the right key chunk and $\hat{k}$ be a wrong candidate. The right leakage predictions $L_{k^\star,i} = \mathrm{Sbox}(k^\star \oplus e_i)$ and the wrong ones $L_{\hat{k},i} = \mathrm{Sbox}(\hat{k} \oplus e_i)$ are linked by the following deterministic relation:

$$L_{\hat{k},i} = \mathrm{Sbox}(\mathrm{Sbox}^{-1}(L_{k^\star,i}) \oplus k^\star \oplus \hat{k}) \,,$$

implying that, if a statistical dependence exists between the random variables $L_{k^\star}$ and $\vec{X}$, then $L_{\hat{k}}$ and $\vec{X}$ are statistically dependent, as well.

Among the most popular side-channel distinguishers, many looks only for linear dependencies, *e.g.* the Difference of Means (DoM) and the Correlation Power Analysis (CPA). The DoM is the one exploited in in [KJJ99] with a mono-bit model (implying that L_k takes only two values, 0 and 1), then generalised for other leakage models in [RO05], under the name of Sum of Differences (SoD). It has the following form:

$$\Delta_k^{DoM} = \hat{\mathbb{E}}[\vec{X} \mid L_k = 0] - \hat{\mathbb{E}}[\vec{X} \mid L_k = 1] \,. \tag{2.11}$$

The formulation of the SoD may be found in Sec. 2.5.2 in the context of profiling attacks. More generally, when the right key candidate is known (*e.g.* the open sample's key in a profiling context), every distinguisher may be used as an indicator of the position of the traces' time samples that mostly contribute to the dependency detection.

The CPA distinguisher, proposed by **[brier2004correlation]**, also detects linear dependencies. It exploits an estimation $\hat{\rho}$ of Pearson correlation-coefficient: $\Delta_k^{CPA} = \hat{\rho}(\vec{X}, L_k)$.

Other kinds of more general distinguishers (*e.g.* the Mutual Information Analysis (MIA) **[gierlichs2008mutual**; **batina2011mutual]** and the Kolmogorov-Smirnov test-based ones (KS) **[veyrat2009mutual]**) look for a wider range of dependencies. With the KS distinguisher, the probability distributions of $\vec{X}$ and L_k are globally compared, and the key for which the two distributions looks more close to each other is selected. The MIA distinguisher consists in an estimation of mutual information between $\vec{X}$ and L_k: $\Delta_k^{MIA} = \hat{I}(\vec{X}, L_k)$. It is an information-theoretic measure that

expresses the quantity of information one has obtained on $\vec{X}$ by observing L_k. The great generality of the MIA distinguisher comes at the cost of two drawbacks. First, a considerable practical inefficiency, due to the fact that the computation of the mutual information requires the estimation of some continuous probability densities, which requires in turn a considerable amount of attack traces. Second, in relation to how anticipated in Rem. 2.1, the MIA distinguisher, if provided with some perfect probability densities estimations, is by definition prone to identify statistically dependence between wrong key hypothesis and actual leakages, leading to unsuccessful attacks, unable to distinguish the right hypothesis among the wrong ones.

Finally, a last widely exploited distinguisher is the Maximum Likelihood one (ML) [CRR03], sometimes referred to as Bayesian distinguisher [**mangard2011one**]. It selects the key candidate that better explains the observed acquisition, in terms of probability: $\Delta_k^{ML} = \Pr(\vec{X} \mid L_k)$. It is the optimal distinguisher, in the sense that it maximizes the probability of a successful attack [**heuser2014good**]. The optimality comes at the cost of the requirement for the knowledge of the conditional probability distribution $\Pr(\vec{X} \mid L_k)$, which can only be estimated *via* a preliminary profiling phase. Indeed, this distinguisher is only available in profiling attacks.

Various works in literature have proposed comparison among the common distinguishers. For instance, Doget *et al.* [**doget2011univariate**] show that some distinguishers are equivalent among them, in the sense that are obtained by a same distinguisher under different leakage models. Mangard *et al.* [**mangard2011one**] showed that, even when fed with the same leakage model, some classical different distinguishers (in particular the CPA and the ML ones) performed in the same way (in terms of success rate) when the noise variance of the acquisitions is sufficiently high. Finally, Heuser *et al.* [**heuser2014good**] exploited a communication theory flavoured side-channel modelisation, and specified in which special and unrealistic contexts the common distinguishers introduced below were equivalent to the optimal one.

2.5 Profiling Side-Channel Attacks

A profiling attack is divided into two distinct phases. The first one, called *profiling phase* or *characterisation* phase exploits so-called *profiling traces* to build a model of the leakages. Profiling traces are acquisitions taken under known values for the sensitive variable Z, so are couples $(\vec{x}_i, z_i)_{i=1,\ldots,N_p}$ for which the correct association trace/sensitive variable is known. The second phase of a profiling attack is the proper *attack phase*, during which the attacker observes a new set of acquisitions,

under unknown secret key, and takes advantage of the previous characterisation to infer over it. Throughout this thesis, and each time a profiling attack scenario is supposed, we will refer to elements of $\mathcal{Z}$ as *labels*, each one identifying a *class* of traces. We will say that acquired traces associated to a same value $s \in \mathcal{Z}$ *belong* to the same class, identified by the label s. We will say as well that such traces are *labelled* by the value s. By abuse we will also refer to the class s to denote the class of traces labelled by s. In such a context N_s will denote the number of profiling traces belonging to the class s.

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

The profiling phase of a profiling attack may be exploited to estimate from data the leakage model L, as a preliminary step for an attack based over any distinguisher. Anyway, in a profiling attack the optimal attack distinguisher is the ML one, which is the one that leads to the Template Attack introduced hereafter, that will be to us the unique approach we will consider to perform a profiling attack.

2.5.1 Template Attack

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

$$p_{\vec{X} \mid Z=z}(\vec{x}) \,, \tag{2.12}$$

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

$$p_{Z \mid \vec{X}=\vec{x}}(z) = \frac{p_{\vec{X} \mid Z=z}(\vec{x}) p_Z(z)}{p_{\vec{X}}(\vec{x})} \,. \tag{2.13}$$

In the attack phase the attacker acquires new traces that he only can associate to the public parameter E, obtaining couples $(\vec{x}_i, e_i)_{i=1,\dots,N_a}$. Then he makes key hypothesis $k \in \mathcal{K}$ and, making the assumption that each acquisition is an independent observation of $\vec{X}$, he associates to each hypothesis $k \in \mathcal{K}$ a score d_k given by the joint *a-posteriori* probability that follows, and that he computes exploiting estimates (2.13):

$$d_k = \prod_{i=1}^{N_a} p_{Z \mid \vec{X}=\vec{x}_i}(f(k, e_i)) \,. \tag{2.14}$$

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

$$\hat{k} = \underset{k}{\operatorname{argmax}}\, d_k \,. \tag{2.15}$$

Remark 2.2. Since the marginal probability density $p_{\vec{X}}(\vec{x}_i)$ of (2.13) does not depend on key hypothesis, it is usually neglected. Moreover, in many cases the variable Z follows a uniform distribution, so its probability mass function $p_Z(z)$ appearing in (2.13) does not influence the ranking of key hypothesis. It is often neglected as well.

Remark 2.3. In the special case of a simple attack, *i.e.* $N_a = 1$, in which $Z = K$, the problem becomes a classical machine learning classification problem (as we will discuss over in Chapter 3): the attacker wants to classify the unique attack trace, *i.e.* assign to it a class label (the key). In such a case, the choice proposed by (2.15) is known as *Bayes (optimal) classifier*.[4] It is proven to be the optimal choice to reduce the misclassification error [Bis06].

This approach has the theoretical optimality that comes from the maximum likelihood criterion. The crucial point is the estimation of the class-conditional densities (2.12): the efficiency of the attack strongly depends on the quality of such estimates.

2.5.1.1 The Gaussian Hypothesis.

A well-established choice to construct class-conditional densities estimations 2.12 is the one applied in Gaussian TA [CRR03]: it consists in making a class-conditional

[4] 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].

multivariate Gaussian distribution assumption

$$\vec{X} \mid Z = z \sim \mathcal{N}(\vec{\mu}_z, \Sigma_z) \,, \tag{2.16}$$

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

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

Many options and choices influence the implementation of a TA: the suppression or not of the marginal densities in (2.13), the use of the unbiased estimator or the maximum likelihood estimator for the covariance matrices, the addition of an *homoscedasticity* assumption (assume that all class-covariance matrices are equal). This last assumption, proposed in 2014 in SCA literature [CK14b], allows exploiting all profiling traces to estimate a unique so-called *pooled* covariance matrix, instead of using traces belonging to each class to estimate each covariance matrix separately. The pooled estimation gains in accuracy.

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

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

2.5.2 Points of Interest and Dimensionality Reduction

Side channel traces are usually acquired by oscilloscopes with a very high sampling rate, which permits a powerful inspection of the component behaviour, but at the same time produces huge-dimensional data, consisting in thousands, or even millions of points. Nevertheless, often only a relatively small part of these time samples is informative, i.e. statistically depends, independently or jointly, on a sensitive target variable. These informative points are called *Points of Interest* (PoI). The dimensionality reduction of the traces is a fundamental pre-processing phase to get efficient and effective SCAs, not too expensive in terms of memory and time consumption. The problem of performing an opportune dimensionality reduction goes hand in hand with the research of PoIs: a convenient dimensionality reduction should enhance the contribution of such PoIs while reducing or nullifying the one provided by non-interesting points. The goal of researches in this context is to study and develop techniques to characterise PoIs and to apply convenient dimensionality reduction techniques, that allow reducing the size of the acquisitions

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

$$\vec{\mu}_s = \hat{\mathbb{E}}[\vec{X} \mid Z = s] = \frac{1}{N_s} \sum_{i\colon z_i = s} \vec{x}_i \,. \tag{2.17}$$

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

$$\vec{\varrho}_s = \hat{\mathrm{Var}}(\vec{X} \mid Z = s) = \frac{1}{N_s - 1} \sum_{i\colon z_i = z} (\vec{x}_i - \vec{\mu}_s)^2 \,, \tag{2.18}$$

where the estimation of the variance $\hat{\mathrm{Var}}$ of a vector has to be intended entry-wise. 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.

2.6 Main Side-Channel Countermeasures

To counteract SCAs, strategies that aim at making leakages independent from the processed sensitive data have to be implemented. We can distinguish two broad

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

Name of the estimate	Definition
SOD	$\tau(t) = \sum_{\substack{z_1,z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\vec{\mu}_{z_1}(t) - \vec{\mu}_{z_2}(t))$
SOSD	$\tau(t) = \sum_{\substack{z_1,z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\vec{\mu}_{z_1}(t) - \vec{\mu}_{z_2}(t))^2$
SOST (version [GLRP06])	$\tau(t) = \frac{\sum_{\substack{z_1,z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\vec{\mu}_{z_1}(t) - \vec{\mu}_{z_2}(t))^2}{\frac{\vec{\varrho}_{z_1}}{N_{z_1}} + \frac{\vec{\varrho}_{z_2}}{N_{z_2}}}$
SOST (version [BDP10])	$\tau(t) = \frac{\sum_{\substack{z_1,z_2 \in \mathcal{Z} \\ z_1 \neq z_2}} (\vec{\mu}_{z_1}(t) - \vec{\mu}_{z_2}(t))^2}{\vec{\varrho}_{z_1} + \vec{\varrho}_{z_2}}$
SNR	$\tau(t) = \frac{\hat{\mathrm{Var}}(\vec{\mu}_Z(t))}{\hat{\mathbb{E}}[\vec{\varrho}_Z(t)]}$ (2.19)

groups of such countermeasures: those that aim at hiding the data and those that are designed to mask the data. The two approaches may even be combined.

2.6.1 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,[5] 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

[5]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 [**popp2005masked**].

may also be combined.

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

2.6.2 Masking

Masking countermeasures derive from the idea of applying secret-sharing methods to counteract side-channel attacks. Secret-sharing methods consist in strategies to distribute a secret message amongst a group of participants. Each participant receives a piece of information, called *share* and the original message can only be reconstructed if a sufficient number of participants collaborate, putting in common the knowledge of a sufficient number of shares. The idea of applying secret-sharing to counteract SCAs was first proposed by Chari *et al.* [Cha+99] and Goubin and Patarin [**goubin1999and**]. 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 [**messerges2000securing**; **akkar2001implementation**; **blomer2004provably**; **oswald2005side**; **schramm2006higher**; **rivain2010provably**; **moradi2011pushing**; **coron2013higher**; **bilgin2014more**; **de2015higher**; **goudarzi2017fast**; **journault2017very**; ISW03]. 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

$$Z = M_1 \star \cdots \star M_d \tag{2.20}$$

holds for a group operation $\star$ (*e.g.* the exclusive or for the most popular Boolean masking already proposed in the seminal papers [**goubin1999and**; Cha+99]). 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.[6] This fact was enlighten by Chari *et al.* in 1999 [Cha+99], then complemented by Prouff and Rivain in 2013 [PR13]. As a consequence of such an exponential complexity behaviour, the number d of shares plays the role of a security parameter for a masking scheme and the method is usually referred to as $(d-1)$th-order masking (since it involves

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

$(d-1)$ random values, called *masks* and one value determined by the sensitive variable and the relation (**??**), 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 dth-order attack has to be run. In the first literature about HO-SCA (for instance [**messerges2000using**; **Waddle2004**; **joye2005second**; **oswald2006practical**] 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 [**peeters2005improved**; **standaert2005masking**]. So the definition of dth-order SCA has mutated in time (see for instance a different formalization in [**piret2008security**]). Today it is most-widely accepted to define a dth-order attack as an attack that looks for key-discriminant information in some dth-order statistical moment of the signal, while the number of time samples of the signals that participate to such a statistic defines the *multivariability* of the attack [**gierlichs2010revisiting**; **batina2011mutual**; Car+14]. For example a 2nd-order attack against a parallel implementation may be univariate if a single time sample is used to derive key-dependent information. In general for attacks against software implementations, a dth-order attack is usually d-variate. In such a case the research of interesting d-tuples of time samples still raises the complexity of the attacks. Even in the favourable case in which a profiling attack is allowed, two cases must be distinguished: the attacker has or not access to the masks values during profiling. In the former case the attacker can use the shares as target sensitive variables during the profiling phase, looking for PoIs for each one of them. Thus, the PoI research complexity grows only linearly with the number d of shares. In the latter case the attacker cannot infer independently over each share and classical tools for PoIs research are inefficient. This issue is the main motivation that leads us to consider

solutions based over the Kernel Discriminant Analysis (KDA) tool, as we will discuss in Chapter 5.

Chapter 3

Introduction to Machine Learning

3.1 Basic Concepts of Machine Learning

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

3.1.1 The Task, the Performance and the Experience

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

- *Regression:* the computer is asked to approximate a mapping function from some input variables to some continuous output variables, *e.g.* approximate $F \colon \mathbb{R}^D \to \mathbb{R}$.

- *Classification:* the computer is asked to specify which class or category an input belongs to, being $\mathcal{Z}$ the set of the possible classes. The learning algorithm is thus asked to construct a function $F \colon \mathbb{R}^D \to \mathcal{Z}$. We remark that this task is similar to the regression one, except for the form of the output, since in general

$\mathcal{Z}$ is a discrete finite set, and not continuous. A slightly variant solution to the classification task consists in constructing a function $F\colon \mathbb{R}^D \to \{0,1\}^{|\mathcal{Z}|}$, if elements of $\mathcal{Z}$ are expressed *via* the *one-hot encoding* (see 2.1). A variant of the classification task consists in finding a function F defining a probability distribution over classes.

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

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

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

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

- the *supervised* learning algorithms access to a dataset of examples, each associated in general to a *target* or *label*. The term supervised reflects the fact that the learning is somehow guided by an instructor that knows the right answer over the learning dataset;

- the *unsupervised* learning algorithms access to a dataset, without any associated target. They try to learn useful properties of the structure of the dataset.

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

3.1.2 Example of Linear Regression

The regression task is not of high interest for the rest of this thesis, but is the most direct example to keep in mind to understand some basic Machine Learning concepts, such as the underfitting and the overfitting (see 3.1.4). Let us introduce a linear regression model to tackle the regression task: we want to construct a linear function $F\colon \mathbb{R}^D \to \mathbb{R}$, that takes an input $\vec{x}$ and outputs $\hat{y} = \vec{w}^\intercal \vec{x}$, where $\vec{w} \in \mathbb{R}^D$ is a vector of *parameters* that have to be learned by a learning algorithm in order to well describe some data.[1] Let $\mathcal{D}_\cdot = (\vec{\mathcal{X}}_\cdot, \mathcal{Y}_\cdot)$ denote a dataset, where $\cdot$ can stand for train or test depending on the role of the dataset in the experience, and let $|\mathcal{D}|$ denote the size of the dataset. Let us store the examples contained in $\vec{\mathcal{X}}_\cdot$ into a matrix $\mathbf{M}_\cdot \in \mathbb{R}^{D\times|\mathcal{D}|}$ and the targets contained in $\mathcal{Y}_\cdot$ into a targets vector $\vec{y}_\cdot \in \mathbb{R}^{|\mathcal{D}|}$. Let a learned model predict targets y_i by outputting $\hat{y}_i = \vec{w}^\intercal \vec{x}_i$ and let them be collected in turn into a predicted targets vector $\hat{\vec{y}}_\cdot$. The MSE is given by

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

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

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

optimization problem can be given in closed form, by means of the pseudo-inverse matrix $\mathbf{M}^+$ of $\mathbf{M}_{\text{train}}$, as follows:

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

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

3.1.3 Example of Linear Model for Classification

In this thesis we point out a strict relationship between the profiling SCAs and the classification task in Machine Learning context. For this reason we introduce here a very brief overview of how classically such a task is tackled, by means of linear models.
Classifying means assigning a label $z \in \mathcal{Z}$ to an example $\vec{x} \in \mathbb{R}^D$, or equivalently divide the input space $\mathbb{R}^D$ in *decision regions*, whose boundaries are referred to as *decision boundaries*. Making use of a linear model signifies exploiting some hyperplanes as decision boundaries. Datasets whose classes can be separated exactly by linear decision boundaries are said to be *linearly separable*. Following the discussion kept by Bishop in [Bis06], two different approaches to tackle the classification task should be distinguished: the direct research for a discriminant function F that assigns to an example a label, or the prior construction of a probabilistic model. This second approach might in turn be distinguished into two options, depending on whether a generative model (see Sec. 2.5.1), or a discriminative model is constructed (*i.e.* only conditional probability densities of outputs given the inputs are modelled). For this example we consider a probabilistic approach, constructing a generative model. This example will allow on one hand to introduce some interesting functions, such as the *logistic sigmoid* and the *softmax*, that will play a role in the construction of neural networks (see Chapter 6). On the other hand, the example justifies the large exploitation of linear discriminant functions, even in contexts where the goal of the learning algorithm to make decisions directly, dispensing with any probabilistic interpretation. Indeed, linear models come out naturally when adding some assumptions on the data distributions, as those that will be introduced below.

Constructing a generative probabilistic model implies modelling the class-conditional probabilities $p_{\vec{X}}(\vec{x} \mid Z = s_j)$ for $j \in \{1, \ldots, |\mathcal{Z}|\}$ as well as the class priors $p_Z(s_j)$ and $p_{\vec{X}}(\vec{x})$. Let us first consider a 2-class context, *i.e.* $\mathcal{Z} = \{s_1, s_2\}$. Then the posterior

probability for the class s_1 is the following:

$$\Pr(s_1 \mid \vec{x}) = \frac{\Pr(\vec{x} \mid s_1)\Pr(s_1)}{\Pr(\vec{x})} = \tag{3.4}$$

$$= \frac{\Pr(\vec{x} \mid s_1)\Pr(s_1)}{\Pr(\vec{x} \mid s_1)\Pr(s_1) + \Pr(\vec{x} \mid s_2)\Pr(s_2)} \,. \tag{3.5}$$

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

$$a = \log\left[\frac{\Pr(s_1 \mid \vec{x})}{\Pr(s_2 \mid \vec{x})}\right] = \log\left[\frac{\Pr(\vec{x} \mid s_1)\Pr(s_1)}{\Pr(\vec{x} \mid s_2)\Pr(s_2)}\right] . \tag{3.6}$$

Then we might assign the label the class s_1 to $\vec{x}$ if and only if $a > 0$, which corresponds to take as decision boundary the surface defined by $\Pr(\vec{x} \mid s_1)\Pr(s_1) = \Pr(\vec{x} \mid s_2)\Pr(s_2)$. We remark that Eq. (3.4) rewrites as:

$$\Pr(s_1 \mid \vec{x}) = \frac{1}{1 + e^{-a}} = \sigma(a) \,, \tag{3.7}$$

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

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

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

$$\mathbf{a}[j] = \log\left[\Pr(\vec{x} \mid s_j)\Pr(s_j)\right] \,, \tag{3.9}$$

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

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

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

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

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

so that

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

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

$$a = \log\left[\frac{p_Z(s_1)}{p_Z(s_2)}\right] - \frac{1}{2}\mu_1^\intercal\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^\intercal\Sigma^{-1}\mu_2 - \vec{x}^\intercal\Sigma^{-1}(\mu_2 - \mu_1) = \vec{w}^\intercal\vec{x} + w_0, \quad (3.12)$$

where we set

$$\begin{aligned}\vec{w} &= \Sigma^{-1}(\mu_1 - \mu_2)\\ w_0 &= \log\left[\frac{p_Z(s_1)}{p_Z(s_2)}\right] - \frac{1}{2}\mu_1^\intercal\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^\intercal\Sigma^{-1}\mu_2.\end{aligned}$$

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

Assumptions (i) and (ii) also lead to the following expression for the posterior probability for s_1, directly implied by (3.7):

$$\Pr(s_1 \mid \vec{x}) = \sigma(\vec{w}^\intercal\vec{x} + w_0)\,. \quad (3.13)$$

Thus, such a posterior probability is given by the sigmoid acting to a linear function of $\vec{x}$. Similarly, for the multi-class case, the posterior probability of class s_j is given

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

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

3.1.4 Underfitting, Overfitting, Capacity, and Regularization

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

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

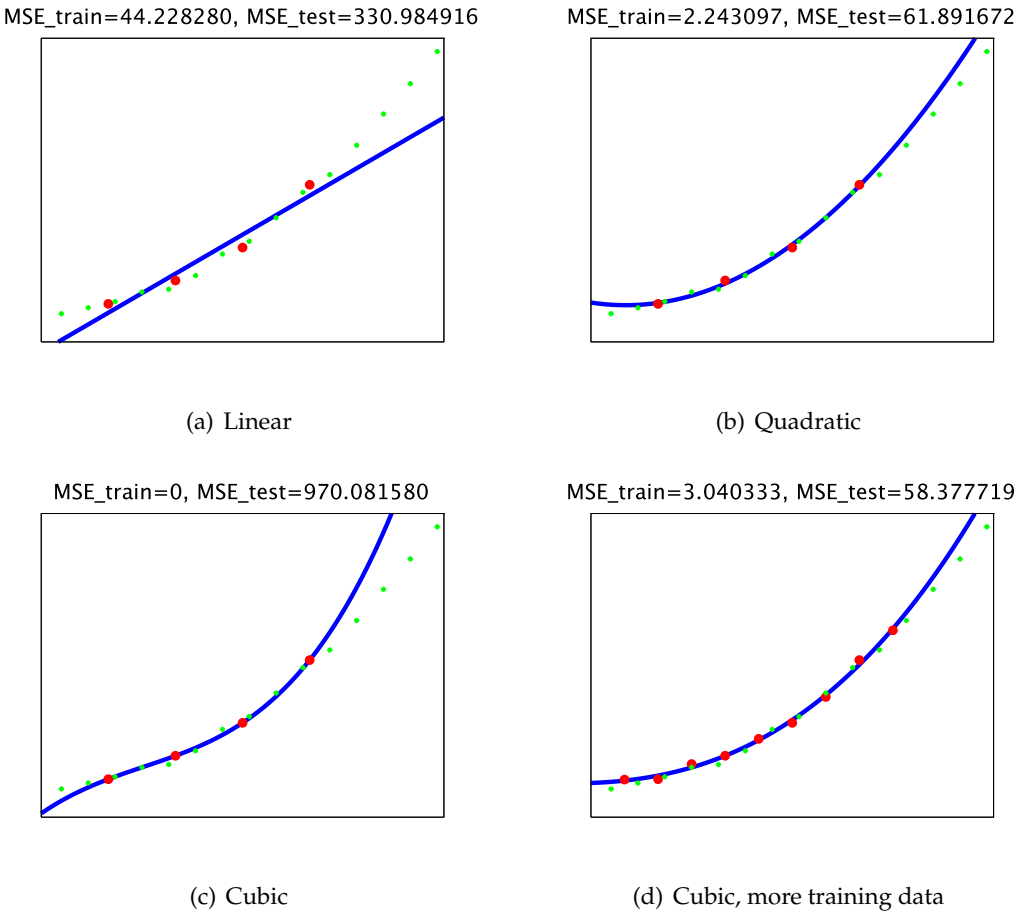
The polynomial regression provides a striking example to understand the underfitting and overfitting phenomena. Consider a problem in which the examples $(x_i, y_i)_{i=1,\dots,N}$ lies in $\mathbb{R} \times \mathbb{R}$ and the true underlying function is quadratic, perturbed

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

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

A very rough way to have an intuition about the capacity of a model is counting the number of its parameters: the capacity in general grows with the number of parameters. Some formal ways to quantify the capacity of a model have been provided in ML literature. The most well-known is the *Vapnik-Chervonenkis dimension*: it measures the capacity of a classifier as the cardinality of the largest set of points the model can classify without errors, for any possible assignment of labels. In practice, quantifying the capacity of a model, especially for complex models as those constructed by neural networks, is very hard and discouraged. On the other hand, these kinds of quantifications have enabled statistical learning theory to formalise and prove some important intuitions, for example the fact that the generalization gap is upper-bounded by a quantity that grows with the model capacity and that shrinks as the number of training examples increases. In Fig. 3.1(d) we observe how the cubic model used for regression on quadratic distributed data ameliorates its performances and reduces the generalization gap despite its excessive capacity, when trained with more examples. This observation basically justifies on one hand the attitude adopted in the branch of ML called *Deep Learning*, and basically based over multi-layer neural networks, consisting in considering very complex models, having confidence in the big size of the typically considered training sets. On the other hand it justifies the interest of *Data Augmentation* (DA) techniques [SSP+03] to respond to an eventual lack of data. Some DA techniques will be proposed in Chapter 6 for the SCA context.

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



(a) Linear

(b) Quadratic

(c) Cubic

(d) Cubic, more training data

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

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

3.1.5 Hyper-Parameters and Validation

The *hyper-parameters* of a model are all the parameters that are priorly set and that are not learned by the learning algorithm. They define the general form of the model. In the polynomial regression example the model had a single hyper-parameter: the degree of the polynomial. It is evident from the example that such a parameter is somehow forced not to be optimised by the means of the learning algorithm: trying to reduce the training MSE, the algorithm would choose a sufficient high degree to interpolate all training points (typically $N - 1$ if N is the number of training examples). This would cause overfitting, as shown in Fig. 3.1(c). In general among all parameters of a model, the hyper-parameters are chosen as those that can not be learned from data because it would cause overfitting, as in the example, or because they are too difficult to optimise.

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

The validation process just described may strongly depend on the way the training set have been split to create the validation one. In order to avoid to validate a model in a strongly data-dependent way, a slightly different process is encouraged

in machine learning community, named the *cross-validation*, which we describe in Appendix A.

3.1.6 No Free Lunch Theorem

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

3.2 Overview of Machine Learning in Side-Channel Context

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

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

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

Part II

Contributions

Chapter 4

Linear Dimensionality Reduction

In this chapter, we explore solutions for dimensionality reduction of side-channel traces exploiting linear combinations of time samples. The results presented in this chapter have been published in the proceedings of CARDIS 2015 [CDP16a].

4.1 Introduction

Linear dimensionality reduction methods produce a low-dimensional linear mapping of the original high-dimensional data that preserves some original feature of interest. 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 structures in data, denoising or compressing data, extracting meaningful feature spaces, and more. A very complete survey about this great variety of linear dimensionality reduction technique has been published in 2015 by Cunningham and Ghahramani [CG15]. They proposed a generalized optimization framework for all linear dimensionality techniques, survey a dozen different techniques and mention some important extensions such as kernel mappings.

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

i.e. it requires a number of observations (traces) which must be higher than the dimension (size) D of them. In some contexts it might be an excessive requirement, which may become unacceptable in many practical situations where the amount of observations is very limited and the traces size is huge.

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

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

In Sec. 4.2 the classical PCA technique is described, then the previous applications of PCA in SCA context are recalled, highlighting the difference between its unsupervised and supervised declination. Finally our contribution to "the choice of components open issue is described" is described: it is based on the Explained Local Variance (ELV) notion, that we will define and argument in the same section. The

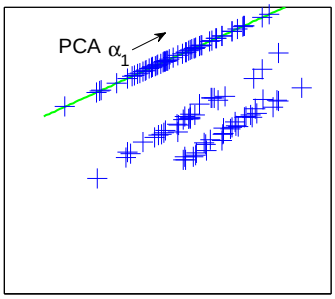


FIGURE 4.1: PCA: some 2-dimensional data (blue crosses) projected into their 1-dimensional principal subspace (represented by the green line).

reasoning behind the ELV selection methodology is essentially based on the observation that, for secure implementations, the leaking information, if existing, is spread over a few time samples of each trace. This observation has already been met by Mavroeidis et al. in [Mav+12], where the authors also proposed a components selection method. As we will see in Sec. 4.2.4, the main difference between their proposal and ours is that in in [Mav+12] the information given by the eigenvalues associated to the PCA components is completely discarded, while the ELV methodology takes advantage of such information as well. We will argue about the generality and the soundness of this methodology and we will 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 due to the SSS problem. The ELV selection tool has been tested in a successive experimental work [CK18]. Unfortunately, the authors of this work could not observe an improvement (nor a worsening) using our new selector, because in their specific case its selection of components were equivalent to the classical one, that will be referred to as EGV in the following.

The LDA technique will be described in Sec. 4.3, together with the description of the SSS problem and some solutions coming from the Pattern and Face Recognition communities [BHK97; Che+00; Hua+02; YY01]. Through some experiments depicted in Sec. 4.4 we will conclude about the effectiveness of the PCA-ELV solution. Finally, in Sec. 4.5 we will experimentally argue about the weakness of all these techniques when data are misaligned.

4.2 Principal Component Analysis

4.2.1 Principles and algorithm description

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

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

$$\mathbf{S} = \frac{1}{N}\mathbf{M}\mathbf{M}^\intercal \, . \tag{4.1}$$

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

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

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

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

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

To compute $\vec{\alpha}_1$ we look for the vector that maximises the variance estimate in (4.3).

The maximisation problem by itself is not well posed, because the variance value is not limited until a restriction is not imposed to the modulo $\|\vec{\alpha}_1| = \sqrt{\vec{\alpha}_1^\intercal \vec{\alpha}_1}$. In order to let the maximization problem have a solution, a restriction is thus imposed: $\vec{\alpha}_1^\intercal \vec{\alpha}_1 = 1$. This constrained optimization problem is handled by making use of Lagrange multipliers:

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

and by computing the partial derivative of Λ with respect to $\vec{\alpha}_1^\intercal$:

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

Thus, stationary points of Λ verify:

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

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

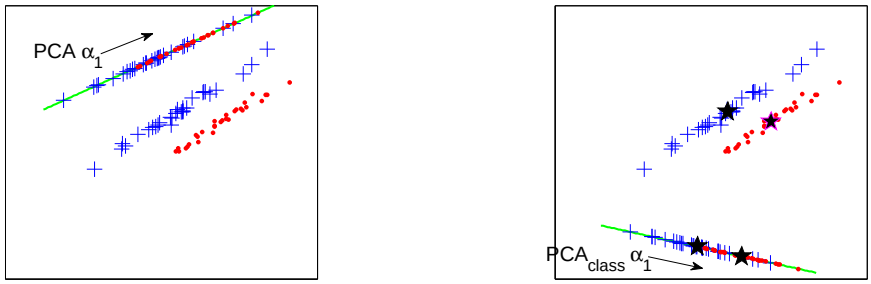
$$\vec{\alpha}_1^\intercal \mathbf{S} \vec{\alpha}_1 = \lambda \vec{\alpha}_1^\intercal \vec{\alpha}_1 = \lambda, \tag{4.7}$$

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

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

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

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



(a) (b)

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

The partial derivatives of (4.8) with respect to $\vec{\alpha}_1^\intercal$ and $\vec{\alpha}_2^\intercal$ are null under the following conditions:

$$\mathbf{S}\vec{\alpha}_1 = \lambda_1\vec{\alpha}_1 \tag{4.9}$$

$$\mathbf{S}\vec{\alpha}_2 = \lambda_2\vec{\alpha}_2\ . \tag{4.10}$$

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

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

4.2.2 Original vs Class-Oriented PCA

The classical version of the PCA method is unsupervised. On the other hand a profiling attacker is not only provided with a set of traces $(\vec{x}_i)_{i=1..N}$, but he also has the knowledge of the target values handled during each acquisition. We denote by $(\vec{x}_i, z_i)_{i=1..N_p}$ the labelled set of traces . In Fig. 4.2 the same data of Fig. 4.1 have been grouped into 2 classes. Even if before projection the two groups are clearly separable, after projecting data over the first principal component given by the classical

PCA algorithm, the separability is lost (Fig.4.2(a). In the supervised context, and for the sake of distinguishing the target value assumed by the target Z in new executions, the idea of the *Class-Oriented* PCA is to consider as *equivalent* all the traces belonging to the same class. Modelling traces of a same class as traces showing the same characteristic form plus a random noise, the denoised characteristic form can be estimated by the sample means of the traces in the class. Let us recall from (2.17) that the empirical mean of traces belonging to the same class s is given by

$$\vec{\mu}_s = \frac{1}{N_s} \sum_{i:\ z_i = s} \vec{x}_i \ ,$$

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

$$\mathbf{S_B} = \sum_{s \in \mathcal{Z}} N_s (\vec{\mu}_s - \overline{\vec{x}})(\vec{\mu}_s - \overline{\vec{x}})^\intercal \ . \tag{4.11}$$

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

4.2.3 Computational Consideration

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

eigenvectors.

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

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

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

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

However, it is not guaranteed that eigenvectors $\vec{\alpha}_i$ obtained in this way have norm equal to 1. This is why a normalization step usually follows.

4.2.4 The Choice of the Principal Components

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

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

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

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

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

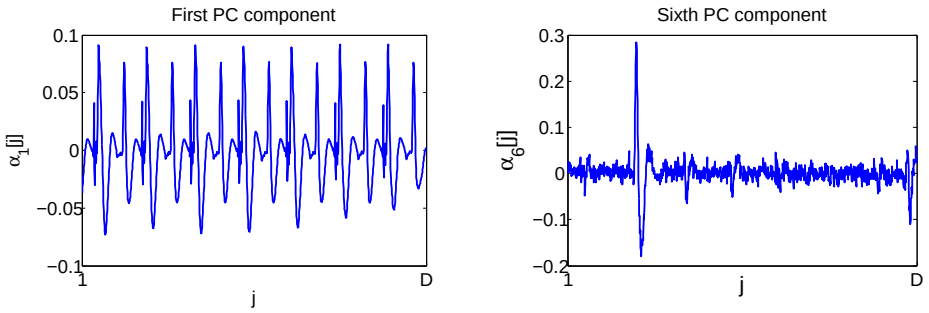


FIGURE 4.3: First and sixth PCs in DPA contest v4 trace set (between time samples 198001 and 199000)

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

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

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

This assumption is reasonable in SCA contexts where the goal of the security developers is to minimize the number of leaking points. Under this assumption, the authors of [Mav+12] use for side-channel attack purposes the *Inverse Participation Ratio* (IPR), a measure widely exploited in Quantum Mechanics domain (see for example [GMGW98]). They propose to use such a score to evaluate the eigenvectors *localization*. It is defined as follows:

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

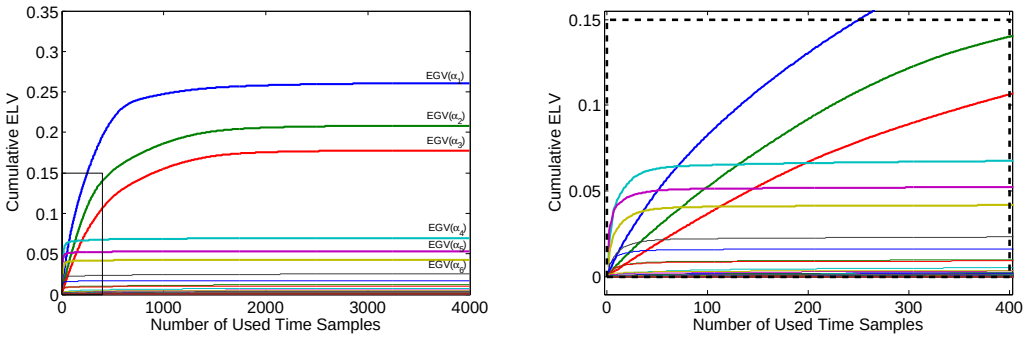


FIGURE 4.4: Cumulative ELV trend of principal components. On the right a zoom of the plot on the left. Data acquisition described in Sec. 4.4.

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 one is based only on the eigenvalues associated to the PCs and does not consider the form of the PCs themselves; the latter completely discards the information given by the eigenvalues of the PCs, considering only the distribution of their coefficients. In the next section we describe a new method, part of the contributions published in [CDP16a], that builds a bridge between the EGV and the IPR approaches. As we will argue, our method, based on the so-called *explained local variance*, does not only lead to the construction of a new selection criterion, but also permits to modify the PCs, choosing individually the coefficients to keep and those to discard.

4.2.4.1 Explained Local Variance Selection Method

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

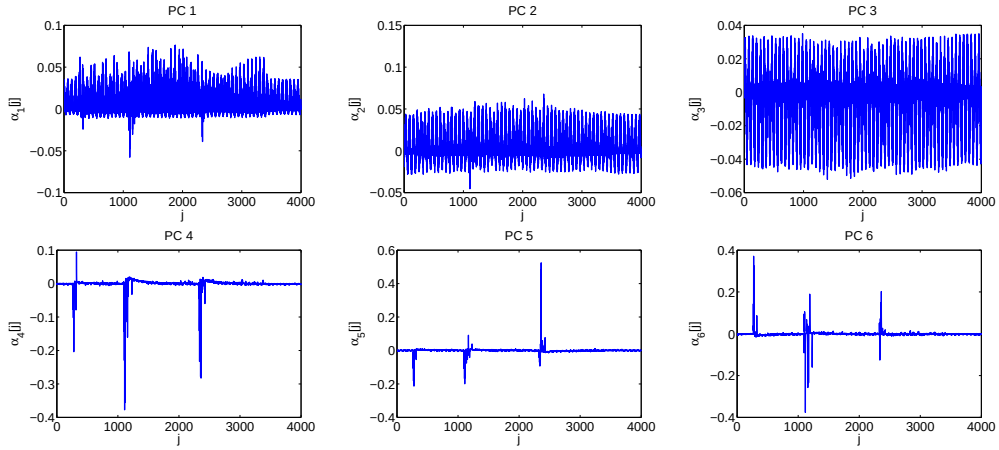


FIGURE 4.5: The first six PCs. Acquisition campaign on an 8-bit AVR Atmega328P (see Sec. 4.4).

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

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

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

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

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

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

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

Let $\mathcal{J} = \{j_1^i, j_2^i, \ldots, j_D^i\} \subset \{1, 2, \ldots, D\}$ be a set of indexes sorted such that $\text{ELV}(\vec{\alpha}_i, j_1^i) \geq \text{ELV}(\vec{\alpha}_i, j_2^i) \geq \cdots \geq \text{ELV}(\vec{\alpha}_i, j_D^i)$. It may be observed that the sum over all the $\text{ELV}(\vec{\alpha}_i, j)$, for $j \in [1, \ldots, D]$, equals $\text{EGV}(\vec{\alpha}_i)$. If we operate such a sum in a cumulative way following the order provided by the sorted set $\mathcal{J}$, we obtain a complete description of the trend followed by the component $\vec{\alpha}_i$ to achieve its EGV. As we can see in Fig. 4.4, where such cumulative ELVs are represented, the first 3 components are much slower in achieving their final EGV, while the 4^{th}, the 5^{th} and

the 6^{th} achieve a large part of their final EGVs very quickly (*i.e.* by adding the ELV contributions of much less time samples). For instance, for $i = 4$, the sum of the $\text{ELV}(\vec{\alpha}_4, j_k^4)$, with $k \in [1, \ldots, 30]$, almost equals $\text{EGV}(\vec{\alpha}_4)$, whereas the same sum for $i = 1$ only achieves about the 15% of $\text{EGV}(\vec{\alpha}_1)$. Actually, the EGV of the 4^{th}, the 5^{th} and the 6^{th} component only essentially depends on a very few time samples. This observation, combined with Assumption 1, suggests that they are more suitable for SCA than the three first ones. To validate this statement, it suffices to look at the form of such components (Fig. 4.5): the leading ones are strongly influenced by the clock, while the latest ones are well localised over the leaking points.

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

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

4.3 Linear Discriminant Analysis

4.3.1 Fisher's Linear Discriminant and Terminology Remark

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

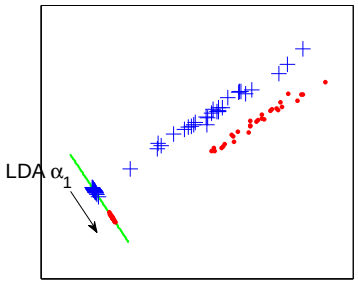


FIGURE 4.6: LDA: some 2-dimensional labelled data (blue crosses and red circles) projected onto their 1-dimensional discriminant component (represented by the green line).

this 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.

4.3.2 Description

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

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

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

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

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

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

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

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

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

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

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

4.3.3 The Small Sample Size Problem

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

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

If the LDA has been introduced relatively lately in the SCA literature, the Pattern Recognition community looks for a solution to the SSS problem at least since the

early nineties. We browsed some of the proposed solutions and chose some of them to introduce, in order to test them over side channel traces.

4.3.3.1 Fisherface Method

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

4.3.3.2 $\mathbf{S_W}$ Null Space Method

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

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

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

4.3.3.3 Direct LDA

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

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

variance assumes the form $\mathbf{S'_W} = {D^\star_B}^{1/2} {V^\star}^\intercal \mathbf{S'_W} V^\star {D^\star_B}^{1/2}$. After storing the C lowest-rank eigenvectors in a matrix $U^\star$, the LDCs obtained via the Direct LDA method are the columns of $V^\star {D^\star_B}^{1/2} {U^\star}^\intercal$.

4.3.3.4 $\mathbf{S_T}$ Spanned Space Method

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

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

4.4 Experimental Results

In this section we compare the different extractors (*i.e.* functions applying a data dimensionality reduction, see Sec. 2.5.2) provided by the PCA and the LDA in association with the different techniques of components selection. Defining an universal criterion to compare the different extractors would not make sense since the latter one should encompass a lot of parameters, sometimes opposite, that vary according to the context (amount of noise, specificity of the information leakage, nature of the side channel, etc.). For this reason we choose to split our comparisons into four scenarios. Each scenario has a single varying parameter that, depending on the attacker context, may wish to be minimized. Hereafter the definition of the four scenarios:

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

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

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

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

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

4.4.1 The testing adversary.

Our testing adversary attacks an 8-bit AVR microprocessor Atmega328P and acquires power-consumption traces via the ChipWhisperer platform [OC14].[2] The target device stores a secret 128-bit key and performs the first steps of an AES: the loading of 16 bytes of the plaintext, the AddRoundKey step and the AES Sbox. It has been programmed twice: two different keys are stored in the device memory during the acquisition of the profiling and of the attack traces, to simulate the situation of two identical devices storing a different secret. The size D of the traces equals 3996. The sensitive target variable is the output of the first Sbox processing, but, since the key is fixed also during the profiling phase, and both Xor and Sbox operations are bijective, we expect to detect three interesting regions (as those highlighted by PCs 4, 5 and 6, in Fig. 4.5): the reading of the first byte of the plaintext, the first AddRoundKey and the first Sbox. We consider an *identity classification* leaking function (i.e. we make minimal assumption on the leakage function, see Sec. **??**), 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. 2.5.1), using C-variate Gaussian templates.

4.4.2 Scenario 1.

To analyse the dependence between the extraction methods presented in Sections 4.2 and 4.3 and the number of attack traces N_a needed to achieve a given GE (Guessing Entropy, see Sec. 2.3), we fixed the other parameters as follows: $N = 50$ ($N_p = 50 \times 256$), $C = 3$ and $\sharp\text{PoI} = 3996$ (all points are allowed to participate in the building of the PCs and of the LDCs). The experimental results, depicted in Fig. 4.7(a)-(b), show that the PCA standard method has very bad performances in SCA, while the

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

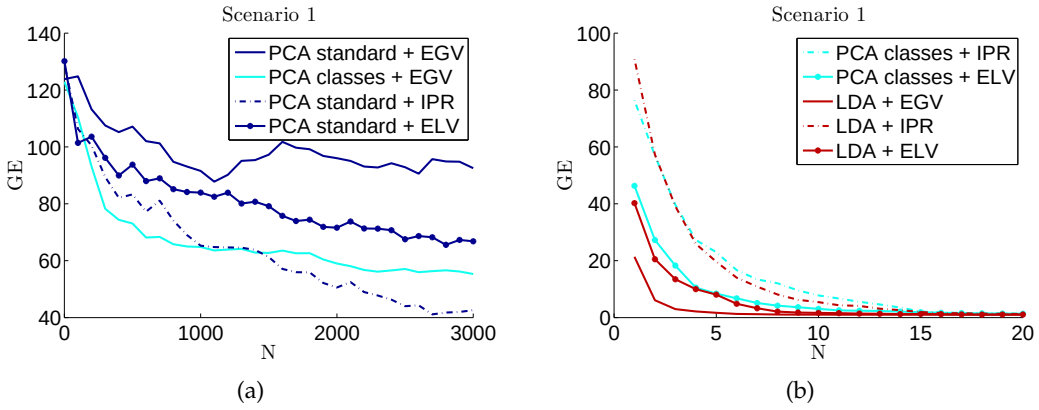


FIGURE 4.7: Guessing Entropy as function of the number of attack traces for different extraction methods. All Guessing Entropies are estimated as the average rank of the right key over 100 independent experiments.

LDA outperforms the others. Concerning the class-oriented PCA, we observe that its performance is close to that of LDA when combined with the selection methods ELV (which performs best) or IPR, while it is similar to the one of classic PCA when associated with the EGV selection.

4.4.3 Scenario 2.

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

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

more profiling traces. It also shows that among class-oriented PCA and LDA, the class-oriented PCA converges faster.

4.4.4 Scenario 3.

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

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

4.4.5 Scenario 4.

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

4.4.6 Overview of this Study and Conclusions

This study focused on two well-known techniques to construct extractors for side-channel traces, the PCA and the LDA. The LDA method is more adequate than the

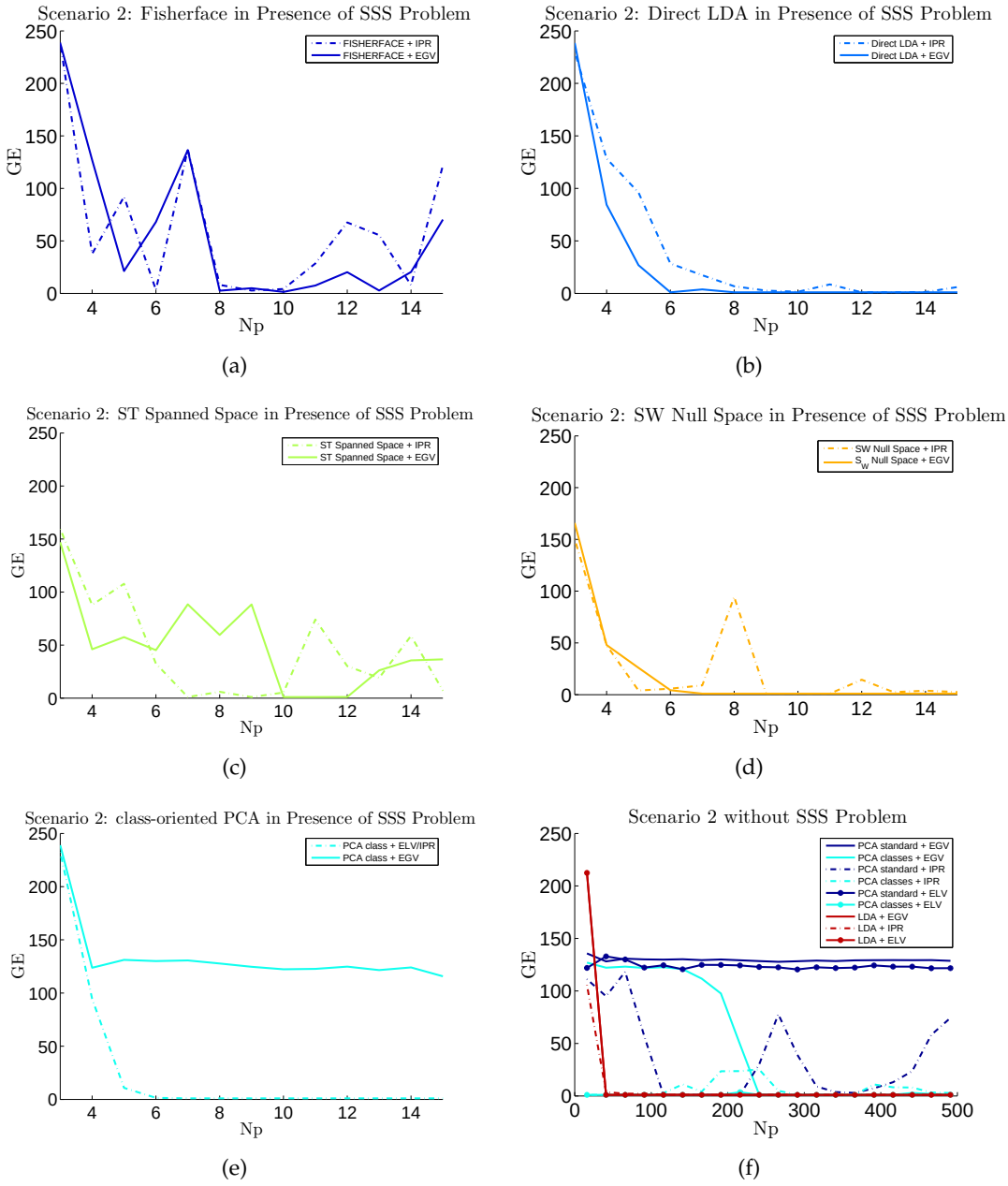


FIGURE 4.8: Guessing Entropy as function of the number of profiling traces. Figures (a)-(d): methods extending the LDA in presence of SSS problem; Figure (e): class-oriented PCA in presence of the SSS problem; Figure (f): number of profiling traces high enough to avoid the SSS problem.

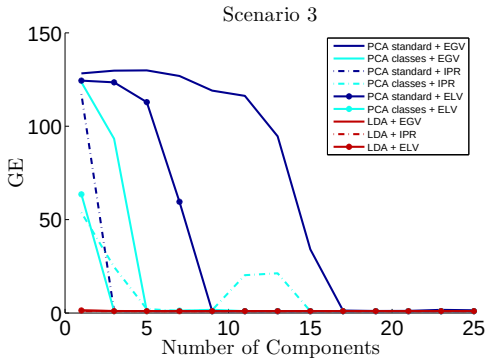


FIGURE 4.9: Guessing Entropy as function of the trace size after reduction.

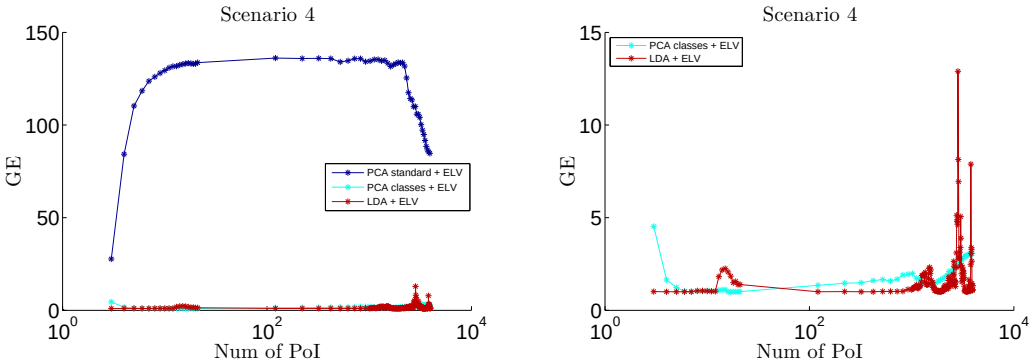


FIGURE 4.10: Guessing Entropy as function of the number of time samples contributing to the extractor computation.

PCA one, thanks to its class-distinguishing asset, but more expensive and not always available in concrete situations. We deduced from a general consideration about side-channel traces, *i.e.* the fact that for secured implementations the vulnerable leakages are concentrated into few points, a new methodology for selecting components, called ELV. We showed that the class-oriented PCA, equipped with the ELV, achieves performances close to those of the LDA, becoming a cheaper and valid alternative to the LDA. Being our core consideration very general in side-channel context, we believe that our results are not case-specific.

A second part of the proposed study analysed experimentally some alternatives to the LDA in presence of SSS problem proposed in Pattern Recognition literature. Such experiments showed that the Direct LDA and the $\mathbf{S_W}$ Null Space Method are promising techniques, exhibiting performances close to the ones given by the ELV-equipped class-oriented PCA. A synthetic overview of the performed comparisons in scenarios 1,2 and 3 is depicted in Table 4.1.

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

TABLE 4.1: Overview of extractors performances in tested situations. Depending on the adversary purpose, given by the parameter to minimize, a ★ denotes the best method, a $+$ denotes a method with performances close to those of the best one and a $-$ is for methods that show lower performances. Techniques introduced in [CDP16a] are highlighted by a grey background.

4.5 Misaligning Effects

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

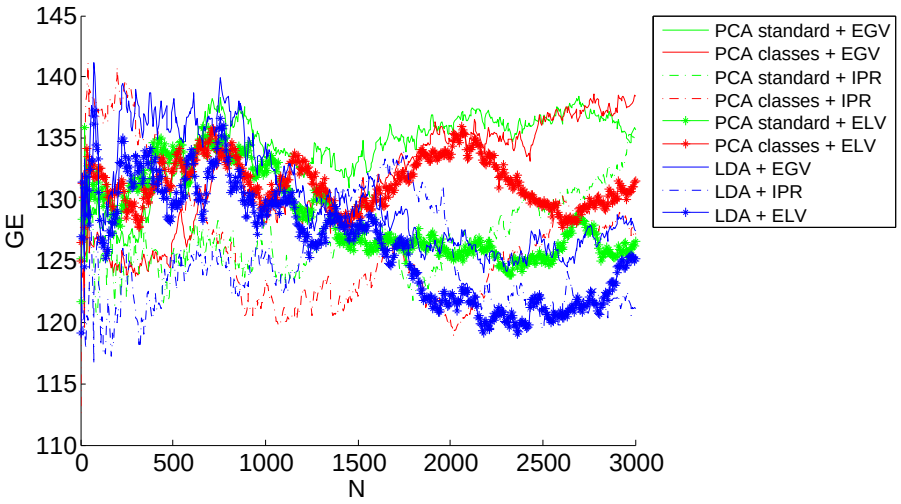


FIGURE 4.11: Degradation of linear-reduction-based template attacks in presence of misalignment.

Chapter 5

Kernel Discriminant Analysis

In this chapter, 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 has 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 search and the linear dimensionality reduction techniques described in the previous chapter are still applicable and efficient. On the contrary, when when the knowledge of the random values is denied, linear techniques are *a priori* inefficient: a non-linear function of the PoIs must 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 context of *profiling attacks without knowledge of the masks*.

5.1.1 Getting information from masked implementations

The SNR estimation defined by (2.1) is an instrument to measure, point by point, the information held by the first-order moment of the acquisition, *i.e.* the information obtainable by observing the variation of the mean of the acquisitions. We refer to such information as a *1st-order information*. In masked implementations, such information is null: at any time sample the mean is independent of Z due to the randomization provided by the shares, namely the quantity $\mathbb{E}[\vec{X}|Z = s]$, seen as a function of s, 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 time samples $(t_1, \ldots, t_d)$ the quantity $\mathbb{E}[\vec{X}[t_1]\vec{X}[t_2]\cdots\vec{X}[t_d]|Z = s]$ (based on a dth-order raw moment) is not constant as a function of s (equivalently, $\mathbb{E}[(\vec{X}[t_1] - \mathbb{E}[\vec{X}[t_1]])\cdots(\vec{X}[t_d] - \mathbb{E}[\vec{X}[t_d]])|Z = s]$ is not constant, using the central moment). We can refer to the information obtainable by observing such variation as a *dth-order information*. In order to let the SNR reveal it, and consequently to get the information being directly exploitable by common attacks, the attacker must preprocess the traces through an extractor ϵ that renders the mean of the extracted data dependent on Z, *i.e.* such that $\mathbb{E}[\epsilon(\vec{X})|Z = s])$ is not constant as a function of s. In this way, the dth-order information is brought back to a 1st-order one.

Property 1 (SCA efficiency necessary condition). Let us assume that Z is represented by a tuple of shares M_i manipulated at d different times. Denoting $t_1, \ldots, t_d$ the unique 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

[1] whence the name *dth-order attacks*
[2] not necessary distinct

variables given by the coordinates of $\vec{X}$ contains at least one term divisible by the the dth-degree monomial $\prod_{i=1,\dots,d} \vec{X}[t_i]$ (see *e.g.* [Car+14] for more information).

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

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

5.1.2 Some strategies to perform higher-order attacks

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 three main approaches were proposed in literature until 2016.

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

The second approach, that avoids the exhaustive enumeration of the d-tuples of time samples, consists in estimating the conditional pdf of the whole region: to this scope, a Gaussian mixture model is proposed in literature [LRP07; Lom+14] and the parameters of such a Gaussian mixture can be estimated through the expectation-maximization (EM) procedure. In [LRP07] 4 variants of the procedure are proposed according to a trade-off between the efficiency and the accuracy of the estimations; the roughest 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 Projection Pursuits (PP) tool for the PoI selection, proposed in [Dur+15] and already introduced in Sec. 4.1, for which we give an outline hereafter. As will be discussed in Sec. 5.4.5, its heuristic nature is the counterpart of the relatively restrained complexity of this tool.

5.1.2.1 Higher-Order Version of Projection Pursuits

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

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

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

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

[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.

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

$$\mathbb{R}^D \xrightarrow{\Phi} \mathcal{F} \underset{\epsilon^{\text{LDA}}}{\overset{\epsilon^{\text{PCA}}}{\rightrightarrows}} \mathbb{R}^C$$

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

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

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

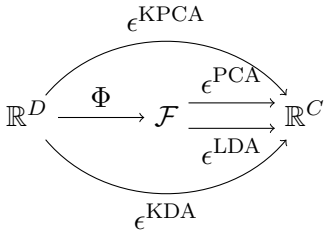


FIGURE 5.2: Applying KDA and KPCA permits to by-pass computations in $\mathcal{F}$.

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* $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$, that has to satisfy the following property, in relation with the function Φ:

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

where $\vec{x}_i$ and $\vec{x}_j$ are data points (*i.e.* side-channel traces) and $\cdot$ denotes 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 $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ that satisfy a convergence condition known as *Mercer's condition* are associated to some map $\Phi : \mathbb{R}^D \to \mathbb{R}^S$, for some S. Importantly, a kernel function is interesting only if it is computable directly from the rough data $\vec{x}$, without evaluating the function Φ.

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

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

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

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

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

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

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

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

In view of the necessary condition exhibited by Property 1, the function $K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j)^d$, hereafter named *dth-degree polynomial kernel function*, is the convenient choice for an attack against implementations protected with $(d-1)$th-order masking. It corresponds to a function Φ that brings the input coordinates into a feature space $\mathcal{F}$ containing all possible d-degree monomials in the coordinates of $\vec{x}$, up to constants. This is, up to constants, exactly the Φ function of (5.2).[5]

5.3 Kernel Discriminant Analysis

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

5.3.1 KDA for dth-order masked side-channel traces

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

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

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

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

where N_s denotes the number of training traces belonging to the class s, $\vec{M}_z$ and $\vec{M}_T$ are two N-sized vectors whose entries are given by:

$$\vec{M}_s[j] = \frac{1}{N_s} \sum_{i:z_i=s} K(\vec{x}_j, \vec{x}_i) \tag{5.8}$$

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

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

$$\mathbf{N} = \sum_{s \in \mathcal{Z}} \mathbf{K}_s (\mathbf{I} - \mathbf{I}_{N_s}) \mathbf{K}_s^\intercal \ , \tag{5.10}$$

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

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

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

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

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

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

For the reasons discussed in Sec. 5.2, the right-hand side of (5.12) may be viewed as an efficient way to process the ℓth coordinate of the vector $\epsilon^{LDA}(\Phi(\vec{x}))[\ell] = \vec{w}_\ell \cdot \Phi(\vec{x})$, without evaluating $\Phi(\vec{x})$. The entries of $\vec{w}_\ell$ are never computed, and will thus be referred to as *implicit coefficients* (see Sec. 5.3.2 below). It may be observed that each discriminant component $\epsilon_\ell^{\text{KDA}}(\cdot)$ depends on the training set $(\vec{x}_i, z_i)_{i=1,\dots,N_t}$, on the kernel function K and on the regularization parameters μ, appearing in (5.11). A further discussion about μ is proposed in Sec. 5.4.2.

5.3.2 The implicit coefficients

As already said, when the dth-degree polynomial kernel function is chosen as kernel function, the KDA operates implicitly in the feature space of all products of d-tuples of time samples. In order to investigate the effect of projecting a new trace $\vec{x}$ over a

component $\epsilon_\ell^{\mathrm{KDA}}(\vec{x})$, we can compute for a small d the implicit coefficients that are assigned to the d-tuples of time samples through (5.12). For $d = 2$ we obtain that in such a feature space the projection is given by the linear combination computed via the coefficients shown below:

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

5.3.3 Computational complexity analysis

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

- $\frac{N_t^2}{2}D$ multiplications, $\frac{N_t^2}{2}(D-1)$ additions and $\frac{N_t^2}{2}D$ 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 $O(N_t^3)$.

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

5.4 Experiments over Atmega328P

5.4.1 Experimental Setup

The target device is an 8-bit AVR microprocessor Atmega328P and we acquired power-consumption traces thanks to the ChipWhisperer platform [OC14].[6] From the acquisitions we extracted some traces composed of 200 time samples, corresponding to 4 clock cycles (see Fig.5.7(a) or 5.7(b) upper parts). Depending on the attack implementation, we ensure that the acquisitions contain either 2,3 or 4 shares respectively for $d = 2, 3$ or 4. The shares satisfy $M_1 \oplus \cdots \oplus M_d = Z$, where Z takes values in $\mathcal{Z} = \mathbb{F}_2^8$ and represents one byte of the output of the first rount SubByte operation in AES: $Z = \mathrm{Sbox}(P \oplus K^\star)$. The goal of the attack is to recover the subkey $K^\star$. The plaintext P is assumed to be known and the M_i are assumed to be unknown random uniform values. The profiling phase is divided in two sub-phases that exploit two distinct datasets. The first sub-phase, that we will refer to as KDA training phase, aims at

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

constructing the dimensionality reduction function by means of KDA algorithm. It exploits a KDA dataset $\mathcal{D}_{\text{train}} = (\vec{\mathcal{X}}_{\text{train}}, \mathcal{Y}_{\text{train}})$ of size $N_t = 8960$. A known fixed subkey is used to acquire such a dataset, the plaintexts have been chosen to balance the number of classes (*e.g.* $N_s = \frac{8960}{256} = 35$ for each $s \in \mathcal{Z} = \{0, \dots 255\}$ when traces are labelled *via* an 8-bit value). We fixed the dimension C at the value 2 (except for the 2-class KDA for which we chose $C = 1$, see Remark 5.3): we therefore tried to build extractors $\epsilon^{\text{KDA}}(\cdot) = (\epsilon_1^{\text{KDA}}(\cdot), \epsilon_2^{\text{KDA}}(\cdot))$ mapping traces of size 200 samples into new traces composed of 2 coordinates.[7] Afterwards, a bivariate Gaussian TA (see Sec. 2.5.1) is run. Such an attack demands for a proper profiling phase, consisting in the estimation of the class-conditional probabilities. It justifies the second profiling sub-phase anticipated above. It will be referred to as profiling phase and will be performed exploiting a distinct profiling dataset $\mathcal{D}_{\text{profiling}} = (\vec{\mathcal{X}}_{\text{profiling}}, \mathcal{Y}_{\text{profiling}})$, collecting $N_{p,s} = 1000$ traces per class (*e.g.* $N_p = 1000 \times 256$ when profiling is done labelling traces by an 8-bit value), under a fixed known key. The choice of not reusing $\mathcal{D}_{\text{train}}$ as profiling dataset for the Gaussian TA has been done in order to reduce the overfitting risk, discussed in general in Sec. 3.1.4 and that will be discussed in the particular case of KDA in Sec. 5.4.2. The extractor ϵ^{KDA}, has a different behaviour when applied to the traces used to train it and to some new traces: this is inevitable since ϵ^{KDA} uses all training traces has its proper parameters (5.12). Thus, a new unobserved profiling set is mandatory in order to obtain an uncorrupted profiling.

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

5.4.2 The Regularization Problem

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

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

When applying such a regularization, the choice of the constant μ is crucial. Beyond the form of the kernel function, μ is the unique hyper-parameter of the model

[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.

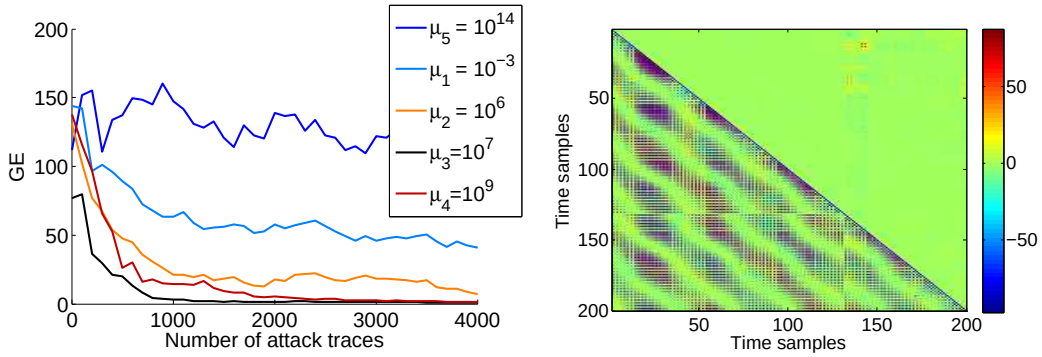
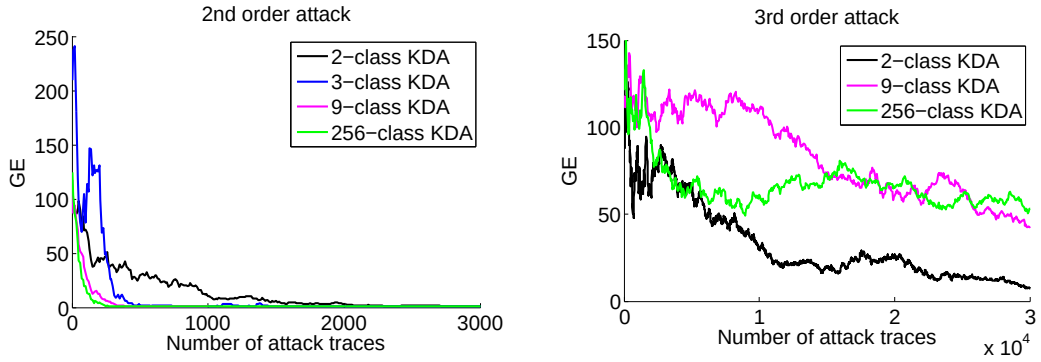


FIGURE 5.3: On the left: template attack guessing entropy vs the number of traces for the attack phase, varying for choices of the constant μ (5.11). On the right: the implicit coefficients assigned to pairs of time samples for μ_3 (upper triangular part) and μ_5 (lower triangular part).

constructed by the KDA algorithm, in the sense explained in Sec. 3.1.5. Its value cannot be learned from data and has to be priorly fixed somehow. For sure it has to be large enough to ensure that $\mathbf{N}$ turns to a positive-definite matrix, but we experimentally observed that the minimal μ for which the positive-definitiveness of $\mathbf{N}$ is attained is often far from being the one that provides a good extractor. In Fig. 5.3 (left) we observe the efficiency of a template attack run in combination with a KDA extractor. The matrix $\mathbf{N}$ is positive-definite for $\mu_1 = 10^{-3}$ but the value that provides the best extractor is much higher (namely $\mu_3 = 10^7$). Still raising the value of μ degrades the quality of the extractor. The right part of Fig. 5.3 shows the implicit coefficients of the extractor (see (5.13)) obtained under μ_3 (upper triangular part) and under μ_5 (lower triangular part). The extractor corresponding to the former one leads to a successful attack and has high values concentrated over the interesting windows, for example $[10..15]$ and $[140..147]$; 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 render the problem computationally stable (which explains why the minimal μ making $\mathbf{N}$ positive-definite may not be a good choice), but also an answer to the overfitting phenomenon. In the case of the KDA the overfitting is observable when ϵ^{KDA} almost perfectly separates the training traces in their classes, while failing in separating the profiling and the attack ones. In [SM99] it is shown that the regularization (5.11) corresponds to the additional requirement for $\vec{\nu}$ to have a small norm $\|\vec{\nu}\|^2$. As every regularization technique, it makes the method less accurate in the learning phase, but in some cases more likely to correctly operate



(a) (b)

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

on new examples.

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

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

5.4.3 The Multi-Class Trade-Off

As discussed in Sec. 4.3, the LDA, and by consequence the KDA, looks for a subspace of the feature space to optimally separate some given classes. The performance of the KDA algorithm raises with the size N_t of the training set. Nevertheless, the number of examples might be bounded by the acquisition context, and even when the N_t can be very high, it may be interesting to minimize it since the KDA complexity is

$O(N_t^3)$. A trade-off must therefore be found between accuracy and efficiency. Assuming that the size of the training set is fixed to N_t, which controls the efficiency, a way to gain in accuracy may be found by appropriately adjusting the number of classes to distinguish: intuitively, the more examples per class, the more accurate the detection of a separating subspace. Then, if the total number of training traces is fixed, in order to raise the number of traces per class, a smaller number of classes must be considered. To do so, a non-injective model $m(\cdot)$ can be introduced, to create a smaller set of labels $m(\mathcal{Z})$ from the initial set $\mathcal{Z}$. The reduced number of classes, *i.e.* the number of labels assigned to the training set after applying the model m, will be denoted by W (it is the cardinality of $m(\mathcal{Z})$). As discussed in Sections **??** and **??**, a widely-accepted power-consumption model for side-channel traces is provided by the Hamming Weight (HW) function, thus we consider and experimentally compare the following models:

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

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

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

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

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

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

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

A balanced training set of size $N_t = 9000$ (instead of 8960) has been used to run the experiments for 2-class, 3-class and 9-class KDA. For the sake of consistency[8] between the pre-processing phase and the attack phase, when a non-injective model is applied to the labels of the training set to reduce the number of classes, the same model is exploited to run the template attack: W templates (one per each class) are estimated from the profiling set (that contains $N_p = W \times 1000$ traces) and compared

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

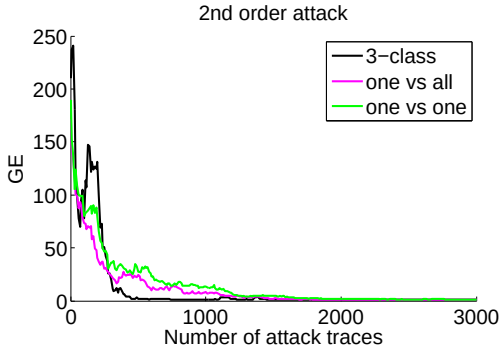


FIGURE 5.5: Performance of template attacks run over 3-class KDA subspaces: multi-class, one vs one and one vs all approaches compared.

to the attack traces. Thus, results of the experimental comparison of these different multi-class approaches depicted in Fig. 5.4 are obtained using different template attacks. It may be remarked that as W decreases the efficiency of the attack is supposed to decrease as well, because each attack trace contributes in distinguishing the right key $K^\star$ only from a growing-size set of indistinguishable hypotheses.

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

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

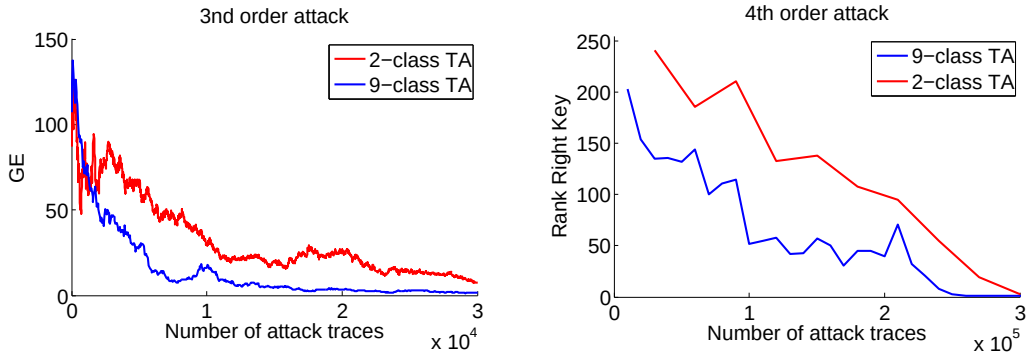


FIGURE 5.6: Left: guessing entropy (over 10 independent tests) for a 2-class and a 9-class 3rd-order template attack. Right: right key rank of a 2-class and a 9-class 4th-order template attack.

is obtained by the 2-classes strategies.[9] We therefore chose to not consider them for the higher-order experiments.

5.4.4 Asymmetric Preprocessing/Attack Approach

In previous section we appealed a consistency principle to justify the choice of running a W-class template attack after a W-class KDA extraction. Here we propose a different reasoning: the consistency principle does not grant that an extractor ϵ^{KDA} trained with W classes is not able to separate W' classes with $W' > W$. As seen in Sec. 5.3.2, an extractor ϵ^{KDA} always implicitly performs a weighed sum, via the implicit coefficients, of centred products of time samples. If ϵ^{KDA} is effective, the implicit coefficients which have the highest magnitude must correspond to time samples corresponding to the manipulation of sensitive data (*e.g.* the variable shares when masking is applied). This property is not necessarily related to the number of classes used to train the extractor.

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

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

5.4.5 Comparison with Projection Pursuits

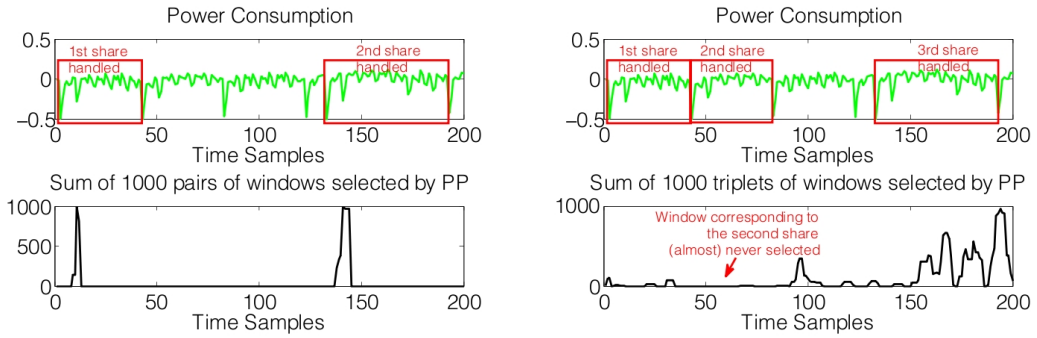
To get a fair comparison, we run the PP algorithm (see Sec. 5.1.2.1) over the same training set used to evaluate the KDA in Sec.5.4. The best results in the 2nd-order context were obtained with the HW model (*i.e.* $|\mathcal{Z}| = 9$). In this case T_{det} is fixed to 0.7. Since 4 training sets are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

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

5.5 Conclusions and Drawbacks

In this chapter we analysed the use of the KDA method to extract small-sized informative features from side-channel acquisitions protected by a $(d-1)$th-order masking countermeasure. The KDA naturally extends the LDA technique to the generic dth-order context. It requires the choice of a so-called kernel function. We

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



(a) (b)

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

proposed to choose a polynomial kernel function, because it perfectly fits the necessary condition to effectively perform a higher-order side-channel attack. Indeed, in this way the obtained extractor provides the linear combination of all possible dth-degree monomial in the time coordinates of the traces, which maximises the SNR. The main obstacle to the problem of PoI selection in higher-order context is that information lies in a space whose dimension grows combinatorially with d, implying computational difficulties. Nevertheless, the KDA only implicitly operates in such a space, by means of a so-called kernel trick, implying that its complexity is independent of the sharing order d. This property represents the main advantage of the KDA. Experiments described in this chapter for 2nd-order, 3rd-order and 4th-order contexts confirmed that our new approach is effective. Anyway, it however presents some drawbacks, discussed hereafter.

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

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

the parameters defining it, together with the entries of the eigenvectors $\vec{\nu}_1, \ldots, \vec{\nu}_Q$. This might be a surprisingly huge number of parameters: for example in our experiments, the extractor $\epsilon^{\mathrm{KDA}} : \mathbb{R}^{200} \rightarrow \mathbb{R}^2$ constructed by exploiting a 8960-sized training set counts $(8960 + 2) \times 200 = 1.792.400$ parameters. In 2nd-order context, this number is much higher than the number of implicit coefficients assigned to all possible 2nd-degree monomials in time samples, which is $\binom{200+2-1}{2} = 20100$.

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

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

not specific to SCA context. Indeed, it is the one that pushed the Machine Learning community to develop the Deep Learning (DL) branch: as we will see in next chapter, and as anticipated in Sec. 1.3.3, DL models are conceived to integrate in a unique optimising process (the learning phase) any preprocessing with the model construction.

Chapter 6

Convolutional Neural Networks

Aiutiamoli a fare da soli!
Let's help them to do themselves!

— Maria Montessori

In this chapter we explore a new strategy to perform profiling SCAs, addressing the misalignment issue and endorsing the Deep Learning (DL) paradigm. To this aim we present results published in [CDP17], where Convolutional Neural Networks are proposed to help against misalignment-oriented countermeasures. Actually, the term Time-Delay Neural Network (TDNN) would be more appropriated than Convolutional Neural Network. Indeed the TDNNs [LWH90] consist in the Convolutional Neural Networks applied to one-dimensional data, as side-channel traces are. Nevertheless, the fame that CNNs reached in last years, and especially since 2012, where a CNN architecture (the "AlexNet") [KSH12] won the *ImageNet Large Scale Visual Recognition Challenge*, a large-impact image recognition contest, leads to the disappearing of term TDNN from DL literature. Today, to specify the architecture of a TDNN in the most common DL libraries, one needs to exploit functionalities related to the CNNs' architecture, specifying *e.g.* that one of the input dimensions equals 1. For these reasons we kept the term CNN for our discussion.

6.1 Motivation

The context we choose to study DL techniques, and CNNs in particular, is the one of cryptographic implementations protected by countermeasures aiming at enhancing misalignment or desynchronisation in side-channel acquisitions. The latter countermeasures are either implemented in hardware (random hardware interruption or non deterministic processors [IPS02; MMS01], unstable clock [Moo+02; Moo+03]) or in software (insertion of random delays through dummy operations [CK09; CK10], shuffling [VC+12]). Techniques analysed in previous chapters were applied in contexts where acquisitions were perfectly synchronous, and are not able to well extend to desynchronised context, as briefly observed in Secs. 4.5 and 5.5.

Desynchronisation might be seen as a noise component of the acquisitions, as done in the leakage model proposed in [Cha+99]. Anyway, it raises the noise that hides sensitive information in the traces. From a statistical point of view, a theoretically satisfying answer to such a noise raise is the solely augmentation of the number of acquisitions: if the attack strategy, in terms of exploited statistical tools, keeps unchanged, increasing the acquisitions by a factor which is somehow linear in the misalignment effect, as discussed in [Man04], might suffice to let the attack be as effective as in the synchronous case. In practice, such an augmentation might be unacceptable for many reasons. First, an attacker or evaluator might have a time or memory bound for the acquisition campaign. Second, the attacked device might implement a security defence denying an unlimited number of executions. Third, attack routines might suffer, in terms of complexity, more than linearly from a raising of the number of data to be treated, *e.g.* the KDA search for a non-linear feature extraction has a complexity that grows in a cubic way with the number of traces.

The second approach proposed in the SCA literature to deal with misaligned trace sets consists in applying a realignment preprocessing before the attack. Two realignment techniques families might be distinguished: a signal-processing oriented one (*e.g.* [Nag+07; WWB11]), more adapted to hardware countermeasures, and a probabilistic-oriented one (*e.g.* [Dur+12]), conceived for the detection of dummy operations, *i.e.* software countermeasures.

We found in Convolutional Neural Networks the possibility of performing a profiling attack in an end-to-end form, directly extracting sensitive information from rough data, without applying any realignment preprocessing. We believe that realignments, a well as dimensionality reduction techniques, as discussed in Sec. 5.5, bring with them the risk of corrupting useful information in data. Indeed a realignment process acts modifying signals with the goal of obtaining some well-synchronised dataset, making traces be somehow similar to each other. On one hand it is not trivial to evaluate the accuracy of a realignment, thus to establish if a performed preprocessing is satisfying. On the other hand, the goal of a realignment is not extracting sensitive and discriminant information from traces. Even if we were able to affirm that a resynchronisation is somehow perfect, by means of some special metrics, nothing guaranties that in the attempt of realigning the trace set the useful information is not discarded. Nowadays, CNNs and DL tools in general are standing out, thanks to their good scalability to "big-data"context. One of their strength is that they are easily parallelisable, and can easily exploit computational facilities as GPUs (or the so-called *TPU - Tensor Processing Units* developed by purpose for NNs), allowing computational accelerations. As we have seen in

Sec. 3.1.4, the higher amount of data are available, the higher capacity is admissible for a ML model, without incurring in overfitting, and higher capacity corresponds to the possibility of learning more complex problems. From this point of view, the success of NNs in last years is mainly due to the always increasing amount of available data, and to their scalability. However, even in contexts where a lack of data may occur, *e.g.* side-channel contexts in which the number of acquisitions may be limited, a stratagem exists in ML literature, under the name of Data Augmentation (DA), that may allow high capacity NNs avoid overfitting and perform well.

6.2 Introduction

Machine Learning approaches often decline in multiple preprocessing phases such as data realignment, feature selections or dimensionality reduction, followed by a final model optimisation. This is the case even for the SCA routines that we considered in previous chapters, or for SCAs that apply realignment preprocessing. Deep Learning is a branch of Machine Learning whose aim is to avoid any preliminary preprocessing step from the model construction work-flow. For example, in Deep Learning the data dimensionality reduction is not necessarily explicitly performed by a distinct learned function ϵ. On the contrary, they directly and implicitly extract interesting features, possibly realign data, and estimate the opportune model to solve the task. The model is searched in a family of models that are composed by a cascade of parametrisable layers, which may be optimised in a single global learning process. Such models are called *Artificial Neural Network*, or simply *Neural Networks* (NNs).

Solution for the KDA Drawbacks

By construction, NNs are the ML answer to the drawback of work-flows we analysed in previous chapters and discussed as *two-phased approach drawback* in last section of Chapter 5. Actually, NNs are answers to other drawbacks pointed out in the same section.

In particular NNs are not memory-based. This implies that, after the training phase whose computational complexity is influenced by the size of the training set, they do not need to access the training set any more. By consequence, the obtained model is in general faster in processing new data, than techniques obtained *via* kernel machines, for which the training traces themselves are part of the model parameters. This property belongs to the characteristics allowing NNs to be easily scalable to huge training sets.

Finally, we pointed out as drawback of techniques analysed in previous chapters their weakness to trace misalignment. Since the CNNs has been developed to treat difficulties as misalignments, scaling, rotations, etc. usually met in image processing, we claim in this chapter, and verify through various experiments, that such CNNs provide an attack strategy that can keeps effective in presence of misalignment countermeasure.

Organisation of the Chapter

In Sections 6.3 and 6.4, notions of DL are introduced. In particular the common classification-oriented *Multi-Layer Perceptron* model is described together with the common practices to train it. The way we exploit NNs to perform SCAs is described in Sec. 6.5, while the performance metrics we will use for experiments are given in Sec. 6.6. A description of the CNN models is provided in Sec. 6.7 while the Data Augmentation techniques that we will exploit are introduced in Sec. 6.8. Finally, three sections are dedicated to experiments. We tested the same CNN architecture against three different targets: in Sec. 6.9.1 we test it against a software countermeasure. In Sec. 6.10 it is tested against a simulated hardware countermeasure, and, in Sec. 6.11, against a real-case cryptographic implementation protected by an enhanced jitter.

6.3 Neural Networks and Multi-Layer Perceptrons

In Chapters 2 and 3 we highlighted a strong analogy between profiling SCAs and classical ML classification task. Thus, we are interested in the NNs' solutions for the classification task. We recall from Chapters 3 that for the classification task, the learning algorithm is asked to construct a function $F\colon \mathbb{R}^D \to \{0,1\}^{|\mathcal{Z}|}$, where elements of $\mathcal{Z}$, *i.e.* the set of classes, are here expressed *via* the *one-hot encoding* (2.1). The output of such a function is said to be *categorical*, *i.e.* $\mathcal{Z}$ is a discrete finite set. A variant of the classification task consists in finding a function $F\colon \mathbb{R}^D \to [0,1]^{|\mathcal{Z}|}$ defining a probability distribution over classes. We will prefer this last formulation. Often for this task, NNs are exploited to create discriminative models, *i.e.* models that directly approximate the latter function F which is actually viewed as the posterior conditional probability of a label given the observed trace. This is the use we propose in this chapter, and it is in opposition to the Template Attack we exploited in previous chapters. Indeed, as described in Sec. 2.5.1, a TA is based over the construction of generative models, *i.e.* the approximation of the *templates*, which coincide with the conditional probabilities of the traces given a label.

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

The *Multi-Layer Perceptrons* (MLPs), or *Feed-forward Neural Networks*, are a family of NN's architectures, associated with a function F that is composed of multiple linear functions and some non-linear functions, called *activations*. The name *feedforward* refers to the fact that the information flows from the input to the output, through the intermediate computations, without any feedback connection in which outputs of the model are fed back into itself. This is in opposition to the so-called *Recurrent Neural Network* structures. The CNNs are a generalisation of the MLPs.

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

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

where:

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

coordinates.

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

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

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

6.4 Learning Algorithm

The weights of an NN are tuned during a training phase. They are first initialized with random values. Afterwards, they are updated *via* an iterative approach which locally applies the (Stochastic) Gradient Descent algorithm [GBC16a] to minimize a loss function quantifying the *classification error* of the function $F(\vec{X})$ over a training set.

6.4.1 Training

The training of an NN is said to be *full batch learning* if the full training database is processed before one update of the weights. At the opposite, if a single training input is processed at a time, then the approach is named *stochastic*. In practice, one

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

often prefers to follow an approach in between, called *mini-batch learning*, and to use small *batches*, *i.e.* groups of training inputs, at a time during the learning. In this case a step of the training consists in:

- selecting a batch of training traces $(\vec{x}_i, z_i)_{i\in I}$ chosen in random order (here I is a random set of indexes),
- computing the outputs, or scores, of the current model function for the input batch $(\vec{y}_i = F(\vec{x}_i))_{i\in I}$,
- evaluating the loss function, which in general involves values $\vec{y}_i$ and z_i
- computing the partial derivatives of the loss function with respect to each trainable weight (this is done through a method called *backpropagation* [LeC+12]),
- updating trainable parameters by subtracting from each a small multiple of the loss gradient (the used multiple is called *learning rate*).

The size of the mini-batch is generally driven by several efficiency/accuracy factors which are *e.g.* discussed in [GBC16b] (*e.g.* optimal use of the multi-core architectures, parallelisation with GPUs, trade-off between regularisation effect and stability, etc.).

An iteration over all the training dataset during the Stochastic Gradient Descent is called an *epoch*. The number of epochs is an important hyper-parameter. Intuitively, running a too low number of epochs may lead to underfitting and high values, while running a too high number of epochs may lead to overfitting. In our experiments, we chose to apply the so-called *early stopping* strategy [Pre12] in order to avoid the need of a prior tuning of the number of epochs. It consists in choosing a stop criterion that will be involved during the training. In general, the choice is done on the basis of a stagnancy or worsen of the validation accuracies or losses across epochs.

6.4.2 Cross-Entropy

The cross-entropy metric is a classical (and often by default) tool to define the *loss function* in a classification-oriented NN [LH05; GBC16a]. It is smooth and decomposable, and therefore amenable to optimisation with standard gradient-based methods. Before providing the definition of cross-entropy in (6.4), we precise the chosen form for the *loss function*. Given a batch of training data $(\vec{x}_i, z_i)_{i\in I}$ and their respective scores returned by the current model $(\vec{y}_i)_{i\in I}$, the *loss function* is defined as the following averaged value:

$$\mathcal{L} = -\frac{1}{|I|} \sum_{i \in I} \sum_{t=1}^{|\mathcal{Z}|} \vec{z_i}[t] \log \vec{y_i}[t] \,, \tag{6.2}$$

where the vector $\vec{z_i}$ denotes the one-hot encoding of the value of the realisation $z_i = s_j$, *i.e.* the vector $\vec{s_j} = (0, \dots, 0, \underbrace{1}_{j}, 0, \dots, 0)$ (as defined in Sec. 2.1). There are two ways to interpret such a choice.

- First, recalling that $\vec{y_i}$ may be interpreted as an estimation of the conditional probability $\Pr[Z \mid \vec{X} = \vec{x_i}]$, the maximum-likelihood principle suggests to drive the training in such a way that for such an estimate the probability of the correct label z_i is as high as possible. Thus, if we suppose that $z_i = s_j$, we want to maximize $\vec{y_i}[j]$ (or equivalently to minimize $-\log \vec{y_i}[j]$).[2] It may be observed that, thanks to the one-hot encoding, in which all entries of $\vec{s_j}$ are null but the jth one, such a log-likelihood rewrites as

$$-\log \vec{y_i}[j] = -\sum_{t=1}^{|\mathcal{Z}|} \vec{z_i}[t] \log \vec{y_i}[t] \,, \tag{6.3}$$

  which equals the quantity averaged in (6.2).

- The second interpretation of the chosen loss function is linked to the fact that it actually represents the average of the cross-entropy of pairs of well-chosen probability mass functions. Indeed interpreting $\vec{z_i} = (0, \dots, 0, \underbrace{1}_{j}, 0, \dots, 0)$ as the pmf of $Z \mid Z = s_j$, which corresponds to the exact probability density we want the network to approximate. Informally speaking, the cross-entropy between two probability distributions, in our case the probability mass functions defined by $\vec{z_i}$ and $\vec{y_i}$, gives a measure of the dissimilarity between them, and is defined as follows:

$$\mathbb{H}(\vec{z_i}, \vec{y_i}) = \mathbb{H}(\vec{z_i}) + D_{KL}(\vec{z_i} || \vec{y_i}) = \mathbb{E}_{\vec{z_i}}[-\log \vec{y_i}] = -\sum_{t=1}^{|\mathcal{Z}|} \vec{z_i}[t] \log \vec{y_i}[t] \,, \tag{6.4}$$

  where $\mathbb{H}$ denotes the entropy and D_{KL} denotes the Kullback-Leibler divergence [Bis06]. Thus, this is an information-theoretic notion, that comes out to be equivalent to the negative log-likelihood formula given by (6.3).

In conclusion, depending on the point of view, minimizing the loss function (6.3), which is a cross-entropy averaged over the traces contained in a batch, corresponds to maximising the likelihood of the right label, or to minimize the dissimilarity between the network estimation of a distribution and the right distribution that we

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

want it to approximate. We chose the loss function (6.2) for our experiments. However, other metrics may be investigated and can potentially lead to better results [MHK10; Son+15].

As justified in Sec. 3.1.5, for the experiments proposed in this chapter we will divide the side-channel profiling set into two subsets: the training one and the validation one. The training set will be processed by batch and used to update the NN's parameters. The validation set is exploited in general at the end of each epoch to monitor the training, and in particular to watch over the incoming of an overfitting phenomenon. Remarkably, cross-validation has not been performed to improve the accuracy of our observation. Instead, we used a side-channel attack set to evaluate both the ability of the trained model to tackle the classification task, and the performance of the obtained attack strategy.

6.5 Attack Strategy with an MLP

The strategy we adopt to perform a SCA, with an MLP, is almost identical to the classical Template Attack described in 2.5.1. The main difference will be that TA is based on generative models, while MLPs are used to construct a discriminative one. Indeed, in TA the templates (2.12) are priorly estimated, while an MLP directly approximates the posterior probabilities (2.13) $F(\vec{x}) \approx p_{Z \mid \vec{X}=\vec{x}}$. Once this approximation is done, the attack strategy proceeds in the same way for both approaches. The attacker acquires the new attack traces, that he only can associate to the public parameter E, obtaining couples $(\vec{x}_i, e_i)_{i=1,\dots,N_a}$. Then he makes key hypotheses $k \in \mathcal{K}$ and, making the assumption that each acquisition is an independent observation of $\vec{X}$, he associates to each hypothesis $k \in \mathcal{K}$ a score d_k given by (2.14), that in terms of MLP model F rewrites as:

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

Finally, the best key candidate $\hat{k}$ is the one maximising the joint probability, as in (2.15)

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

6.6 Performance Estimation

6.6.1 Maximal Accuracies and Confusion Matrix

The accuracy is the most common metric to both monitor and evaluate an NN. As already seen in Sec. 3.1.5, the accuracy is defined as the successful classification rate

reached over a dataset. The *training accuracy*, the *validation accuracy* and the *test accuracy* are the successful classification rates achieved respectively over the training, the validation and the test sets. At the end of each epoch it is useful to compute and to compare the training accuracy and the validation accuracy. For our study, we found interesting to consider the following two additional quantities:

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

In addition to the two quantities above, we will also evaluate the performances of our trained model, by computing a *test accuracy*. Sometimes it is useful to complete this evaluation by looking at the so-called *confusion matrix* (as the one appearing in the bottom part of Fig. 6.7). Indeed the latter matrix enables for the identification of the classes which are confused, in case of misclassification. The confusion matrix corresponds to the distribution over the couples *(true label, predicted label)* directly deduced from the results of the classification on the test set. A test accuracy of 100% corresponds to a diagonal confusion matrix.

6.6.2 Side-Channel-Oriented Metrics

The accuracy metric is perfectly adapted to the machine learning classification problem, but corresponds in side-channel language to the success rate of a Simple Attack, as already discussed in Chapter 2. When the attacker can acquire several traces with varying plaintexts, the accuracy metric is not sufficient alone to evaluate the attack performance. Indeed such a metric only takes into account the label corresponding to the maximal score and does not consider the other ones, whereas an SCA through (6.5) does. To take this remark into account, we will always associate the test accuracy to a side-channel metric defined as the minimal number $N^\star$ of side-channel traces that makes the *guessing entropy* (see 2.3) be permanently equal to 1. In our experiments, we will estimate such a guessing entropy through 10 independent attacks.

6.7 Convolutional Neural Networks

The Convolutional Neural Networks (CNNs) complete the classical MLP model with two additional types of layers, in charge of making them robust to misalignment: the so-called *convolutional* layer based on a convolutional filtering, and the

pooling layer. We describe these two particular layers hereafter.

Convolutional (CONV) layers Convolutional Layers (CONV) are linear layers that share weights across space. A representation is given in Fig. 6.1; since CNNs have been introduced for images [LB+95], such representation differs from the most common one in which layer interfaces are arranged in a 3D-fashion (height, weight and depth). In Fig. 6.1 we show a 2D-CNN (length and depth) adapted to 1D-data as side-channel traces are. To apply a CONV to an input of size $D \times V$, where the initial depth V is one, for 1D-data, n_{filter} small matrices, called *convolutional filters*, of size $W \times V$ (where W is called *kernel size*) are slid over the length dimension of the input by some amount of units, called *stride*. The filters form a window, called *patch* in the Machine Learning language, which defines a linear transformation of $W \times V$ consecutive points of the data into new matrices of size $1 \times n_{\text{filter}}$, arranged in such a way thatn_{filter} is the depth of the layer output. The length dimension of the output of a convolutional layer depends on several parameters: the input length, the stride, and the *padding*. The two most common ways to pad the input are called *same padding* and *valid padding*: with the *same padding* the input is padded with some zeros at the beginning and at the end, in such a way that, for a stride equal to 1, the output has the same length than the input, for a stride equal to 2 the input length is exactly halved, for a stride equal to 3 it is exactly divided by 3, etc. The *valid padding* consists on the contrary to avoid any kind of padding. Only proper data points are used as input, and output length is adjusted: typically, for a stride equal to 1, the output length equals $D - W + 1$, where D is the input length. The coordinates of the window are among the trainable weights of the model. They slid over the input, so they are multiplied by different parts of the datum, but they are constrained to keep unchanged while sliding, *i.e.* to behave in the same way no matter the position of the input entries on the global input datum. This constraint aims to allow the CONV layer to learn shift-invariant features, *i.e.* characteristics of the datum for which the position is not discriminant. Shift-invariant features are largely present in image recognition context, which drove the development of CNNs. For examples the eyes, the nose and the mouth of a person in a picture, are discriminant features for the person no matter their position in the image. The ability at learning shift-invariant features makes CNNs robust to geometrical deformations [LB+95] or to temporal deformation when considering side-channel signals. For this reason they are adequate to counteract misalignment-based countermeasures.

Pooling (POOL) layers In the most typical example of convolutional layer, *i.e.* a layer with stride equal to 1 and *same padding*, the output size equals the input size multiplied by n_{filter}. If many of this kind of convolutional layers are stacked, it leads

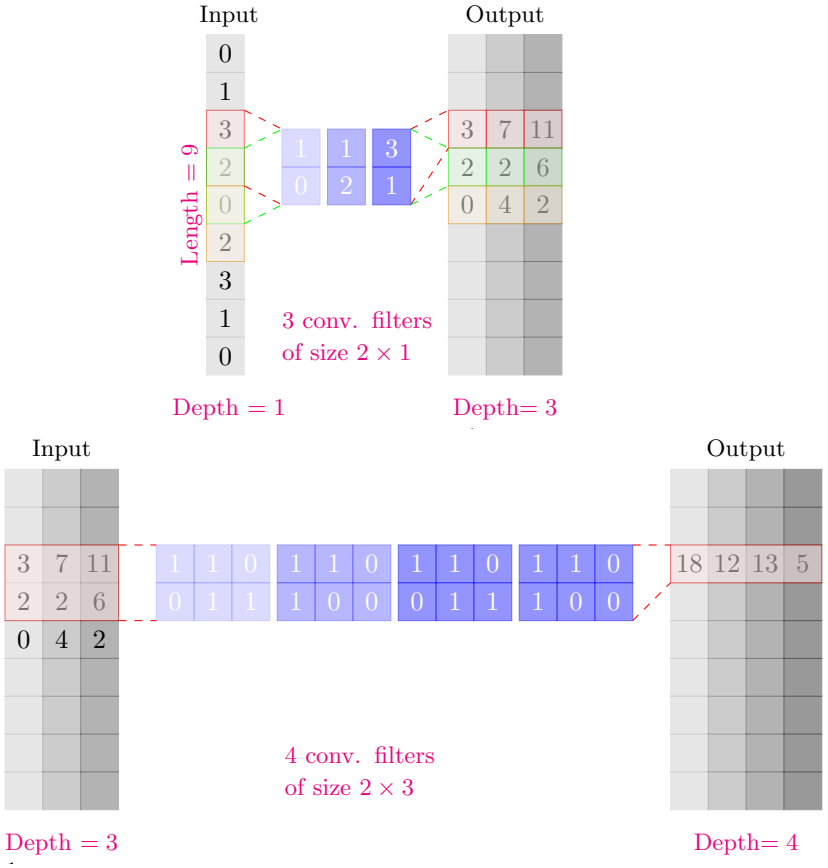


FIGURE 6.1: Two convolutional layers. Top: $W = 2$, $V = 1$, $n_{\text{filter}} = 3$, stride $= 1$. Bottom: $W = 2$, $V = 3$, $n_{\text{filter}} = 4$.

to a complexity exponential growing due to the increasing of data size through layers. To avoid such complexity explosion, the insertion of pooling (POOL) layers is recommended. POOL layers are non-linear layers that reduce the spatial size (see Fig. 6.2). As the CONV layers, they make a filter slide across the input. The filter is 1-dimensional, characterised by a length W, and usually the stride is chosen equal to its length; for example in Fig.6.2 both the length and the stride equal 3, so that the selected segments of the input do not overlap. In contrast with convolutional layers, the pooling filter does not contain trainable weights; it only slides across the input to select a segment, then a pooling function is applied: the most common pooling functions are the *max-pooling*, which outputs the maximum values within the segment, and the *average-pooling*, which outputs the average of the coordinates of the segment.

Discussion The reason why a CONV always applies several filters (*i.e.* $n_{\text{filter}} > 1$) is that we expect each filter to extract a different kind of feature from the input. These extracted features are arranged side-by-side over an additional data dimension, the

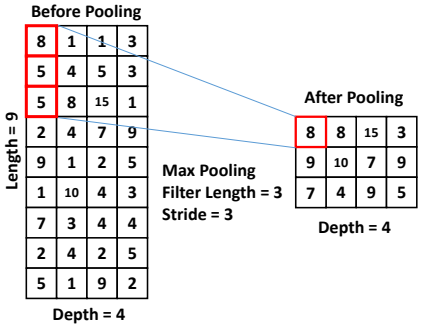


FIGURE 6.2: Max-pooling layer: $W = \text{stride} = 3$.

so-called *depth*.[3] The hope is that during training, automatically, each filter specialises over the detection/recognition/modalisation of a different discriminant feature, and the collection of all discriminant features allows the last network layer concluding a successful classification. As one goes along convolutional layers, higher-level abstraction features are expected to be extracted. The face recognition problem provides a simplified didactic example for this concept: we may think to some first layers' filters that specialise in detecting some local patterns of borders and surfaces. Then we may think to a deeper layer that compose such local features and modelise the angles of eyes' borders. the pupils, their color. Then some deeper layers may compose such feature and modelise the whole eye, which is a more complex feature, and some deeper layers may compose eyes together with noses' features coming from other filters and, going on in this compositional process, modelise the whole face, and assign to it a very abstract feature, *i.e.* the name of the person, which is the goal of the classification task. The fact that many natural data in the works have such a compositional flavour is one of the justifications inventors of CNNs provide to explain the success of such a technique.[4] Actually, analysing and understanding the very first low-level features extracted by a self-trained CNN is a very hard task, and such an impossibility to explain from where discriminant features come out is, in my opinion, one of the characteristics of the DL domain that leads it to be kept unconsidered and disliked by a still quite large community of scientists.

Common architecture The main block of a CNN is a CONV layer γ directly followed by an ACT layer σ. The former locally extracts information from the input thanks to its filters and the latter increases the capacity of the model thanks to its non-linearity. After some $(\sigma \circ \gamma)$ blocks, a POOL layer δ is usually added to reduce the number of neurons: $\delta \circ [\sigma \circ \gamma]^{n_2}$. This new block is repeated in the neural network until obtaining an output of reasonable size. Then, some FCs are introduced

[3] Ambiguity: Neural networks with more that one non-linear layer are called *Deep Neural Networks*, where the *depth* corresponds to the number of layers.

[4] See for example Yann LeCun's class available at https://www.college-de-france.fr/site/yann-lecun/course-2016-02-12-14h30.htm

in order to obtain a global result which depends on the entire input, and not only on local features. To sum-up, a common convolutional network can be characterized by the following formula:[5]

$$s \circ [\lambda]^{n_1} \circ [\delta \circ [\sigma \circ \gamma]^{n_2}]^{n_3}. \tag{6.7}$$

Layer by layer the network increases the spatial depth through convolution filters, adds non-linearity through activation functions and reduces the spatial (or temporal, in the side-channel traces case) size through pooling layers. Once a deep and narrow representation has been obtained, one or more FC layers are connected to it, followed by a softmax function. An example of CNN architecture is represented in Fig. 6.3.

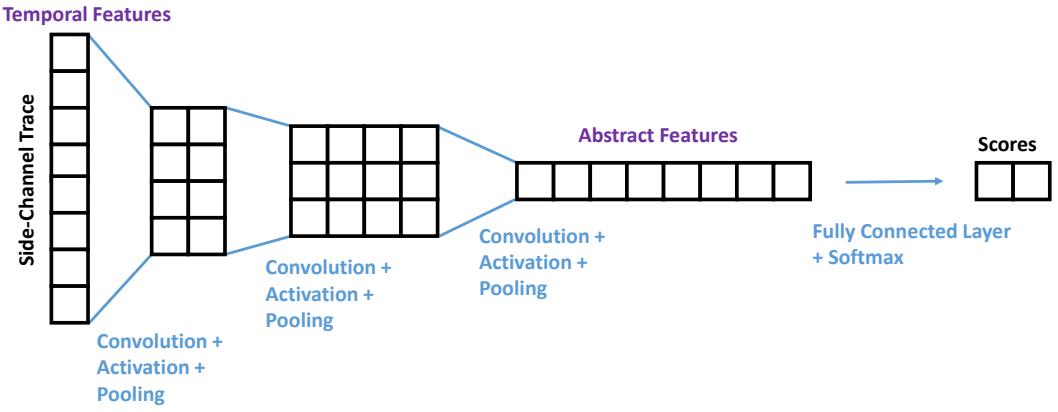


FIGURE 6.3: Common CNN architecture.

6.8 Data Augmentation

As explained in Sec. 3.1.4, ML models are prone to overfitting, especially when their capacity (see Sec. 3.1.4) is very high, as it is often the case with deep networks. Thus, it is sometimes necessary to deal with the overfitting phenomenon, by applying some regularization techniques. As we will see in Secs. 6.9.1 and 6.10 this will be the case in our experiments: indeed we will propose a quite deep CNN architecture, flexible enough to successfully manage the misalignment problems, but trained over some relatively small training sets. This fact, combined with the high capacity of our CNN architecture, implies that the model will *learn by heart* each element of the training set, without catching the truly discriminant features of the traces.

[5] where each layer of the same type appearing in the composition is not to be intended as exactly the same function (*e.g.* with same input/output dimensions), but as a function of the same form.

Instead of applying a proper regularization techniques, we choose to concentrate priorly on the Data Augmentation strategy [SSP+03], mainly for two reasons. First, it is a common practice in side-channel context to increase the number of acquisitions to counteract the misalignment effect. In other terms, misalignment may provoke a "lack of data"phenomenon on adversary's side. In the ML domain, such a lack is classically addressed thanks to the DA technique, and its benefits are widely proved. For example, many image recognition competition winners made use of such a technique (*e.g.* the winner of ILSVRC-2012 [KSH12]). Second, the DA is controllable, meaning that the deformations applied to the data are chosen, thus fully characterized. It is therefore possible to fully determine the addition of complexity induced to the classification problem. In our opinion, other techniques add constraints to the problem in a more implicit and uncontrollable way, *e.g.* the dropout [Hin+12] or the ℓ_2-norm regularization [Bis06].

Data augmentation consists in artificially generating new training traces by deforming those previously acquired. The deformation is done by the application of transformations that preserve the label information (*i.e.* the value of the handled sensitive variable in our context). We choose two kinds of deformations, that we denote by *Shifting* and *Add-Remove*.

Shifting Deformation ($\mathrm{SH}_{T^\star}$) It simulates a random delay effect of maximal amplitude $T^\star$, by randomly selecting a shifting window of the acquired trace, as shown in Fig. 6.4. Let D denote the original size of the traces. We fix the size of the input layer of our CNN to $D' = D - T^\star$. Then the technique $\mathrm{SH}_{T^\star}$ consists (1) in drawing a uniform random $t \in [0, T^\star]$, and (2) in selecting the D'-sized window starting from the t-th point. For our study, we will compare the SH_T technique for different values $T \leq T^\star$, without changing the architecture of the CNN (in particular the input size D'). Notably, $T \lneq T^\star$ implies that $T^\star - T$ time samples will never have the chance to be selected. As we suppose that the information is localized in the central part of the traces, we choose to center the shifting windows, discarding the heads and the tails of the traces (corresponding to the first and the last $\frac{T^\star - T}{2}$ points).

Add-Remove Deformation (AR) It simulates a clock jitter effect (Fig. 6.5). We will denote by AR_R the operation that consists in two steps:

(1) in inserting R time samples, whose positions are chosen uniformly at random and whose values are the arithmetic mean between the previous time sample and the following one,

(2) in suppressing R time samples, chosen uniformly at random.

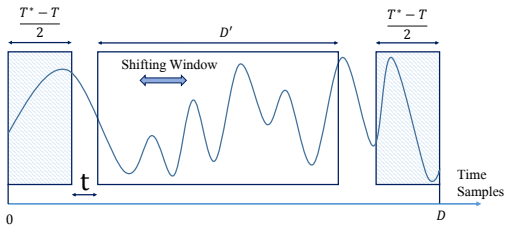


FIGURE 6.4: Shifting technique for DA.

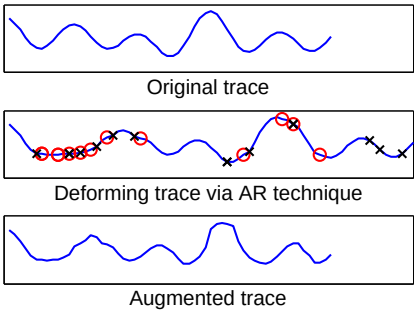


FIGURE 6.5: Add-Remove technique for DA (added points marked by red circles, removed points marked by black crosses).

The two deformations can be composed: we will denote by $\mathrm{SH}_T\mathrm{AR}_R$ the application of a SH_T followed by a AR_R.

Discussion The deformations we propose as Data Augmentation techniques are inspired by the way we modelise the countermeasures' effects. Actually, we propose to turn the misalignment problem into a virtue, enlarging the profiling trace set *via* a random shift of the acquired traces and the AR distortion that together simulate a clock jitter effect. Paradoxically, instead of trying to realign the traces, we propose to further misalign them (a much easier task!). In real-case secure devices evaluation contexts, the acquisition campaign may sometimes represent a bottleneck in terms of time. Further proposals and analyses of DA techniques, maybe inspired by other forms of noise present in side-channel acquisitions, might be interesting tracks for future researches. Actually, the idea of applying DA in profiling side-channel context appeared independently from our work, in another publication in 2017 [Pu+17], under the name of *Trace Augmentation*. In this paper, the augmentation is obtained with a shifting equivalent to our SH deformation, and it is applied as preliminary step for the profiling phase of a Gaussian TA. The authors' goal is to make Gaussian templates more robust to the discrepancy between profiling acquisitions and attack ones. Surprisingly, in the paper, authors observe that this augmentation provides benefits to the attack routine both in case where some discrepancies are present, and in the ideal case. Data Augmentation seems thus to be a good practice independently of the presence or not of specific countermeasures, nor the exploitation or not

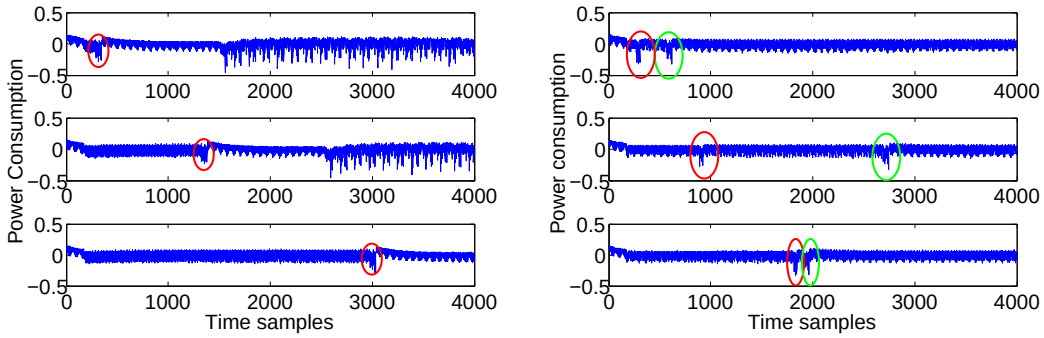


FIGURE 6.6: Left: one leakage protected by single uniform RDI. Right: two leaking operations protected by multiple uniform RDI.

of DL techniques.

6.9 Experiments against Software Countermeasures

In this section we present two preliminary experiments, performed in order to validate the shift-invariance claimed by the CNN architecture, recalled in Sec. 6.7. In the first one, a single leaking operation was observed through the side-channel acquisitions, shifted in time by the insertion of a random number of dummy operations. We will refer to such a countermeasure as Random Delay Interrupt (RDI). In the second one we targeted two leaking operations each delayed by RDI. We remark that this kind of countermeasure is nowadays considered defeated, *e.g.* thanks to resynchronisation by *cross-correlation* [Nag+07]. In this sense, the experiment we present in this section is not expected to be representative of real application cases. The complexity of the state-of-the-art resynchronisation techniques strongly depends on the variability of the shift. When the latter variability is low, *i.e.* when attacks are judged to be applicable, multiple random delays are recommended. It has even been proposed to adapt the probabilistic distributions of the random delays to achieve good compromises between the countermeasure efficiency and the chip performance overhead [CK09; CK10]. Attacks have already been shown even against this multiple-RDI kind of countermeasures, *e.g.* [Dur+12]. The latter attack exploits some Gaussian templates to classify the leakage of each instruction; the classification scores are used to feed a Hidden Markov Model (HMM) that describes the complete chip execution, and the Viterbi algorithm is applied to find the most probable sequence of states for the HMM and to remove the random delays. We remark that this HMM-based attack exploits Gaussian templates to feed the HMM model, and the accuracy of such templates is affected by other misalignment reasons, *e.g.* clock jitter. We believe that our CNN approach proposal for operation classification, is a valuable alternative to

the Gaussian template one, and might even provide benefits to the HMM performances, by *e.g.* improving the robustness of the attack in presence of both RDI and jitter-based countermeasures. This robustness w.r.t. of misalignment caused by the clock jitter will be analysed in Sec. 6.10.

6.9.1 One Leaking Operation

For this experiment, we implemented, on an Atmega328P microprocessor, a uniform RDI [TB07] to protect the leakage produced by a single target operation. Our RDI simply consists in a loop of r *nop* instructions, with r drawn uniformly in $[0, 127]$. Some acquired traces are reported in the left side of Fig. 6.6, the target peak being highlighted with a red ellipse. They are composed of $3,996$ time samples, corresponding to an access to the AES-Sbox look-up table stored in NVM. For the training, we acquired only $1,000$ traces and 700 further traces were acquired as validation data. Our CNN has been trained to classify the traces according to the Hamming weight of the Sbox output; namely, our labels are the nine values taken by $Z = \mathrm{HW}(\mathrm{Sbox}(P \oplus K))$. This choice has been done to let each class contain more than only a few (i.e. about $1,000/256$) training traces. For Atmega328P devices, the Hamming weight is known to be particularly relevant to model the leakage occurring during register writing (see for example Chapters 4 and 5 or [Bel+15]). Since Z is assumed to take nine values and the position of the leakage depends on a random r ranging over 128 values, it is clear that the $1,000$ training traces do not encompass the full $9 \times 128 = 1,152$ possible combinations $(z, r) \in [0, 8] \times [0, 127]$. We undersized the training set by purpose, in order to establish whether the CNN technique, equipped with DA, is able to catch the meaningful shift-invariant features without having been provided with all the possible observations.

For the training of our CNN, we applied the SH_T data augmentation, selecting $T^\star = 500$ and $T \in \{0, 100, T^\star\}$; this implies that the input dimension of our CNN is reduced to $3,496$. Our implementation is based on Keras library [Cho+15] (version 1.2.1), and we run the trainings over an ordinary computer equipped with a gamers market GPU, a GeForce GTS 450. For the CNN architecture, we chose the following structure:

$$s \circ [\lambda]^1 \circ [\delta \circ [\sigma \circ \gamma]^1]^4, \tag{6.8}$$

i.e. (6.7) with $n_1 = n_2 = 1$ and $n_3 = 4$. To accelerate the training we applied a technique proposed in 2015 [IS15], consisting in the introduction of a so-called *Batch Normalization* layer [IS15] after each pooling δ. The network transforms the $3,496 \times 1$ inputs in a 1×256 list of abstract features, before entering the last FC layer $\lambda : \mathbb{R}^{256} \to \mathbb{R}^9$. Even if the ReLU activation function [NH10] is classically

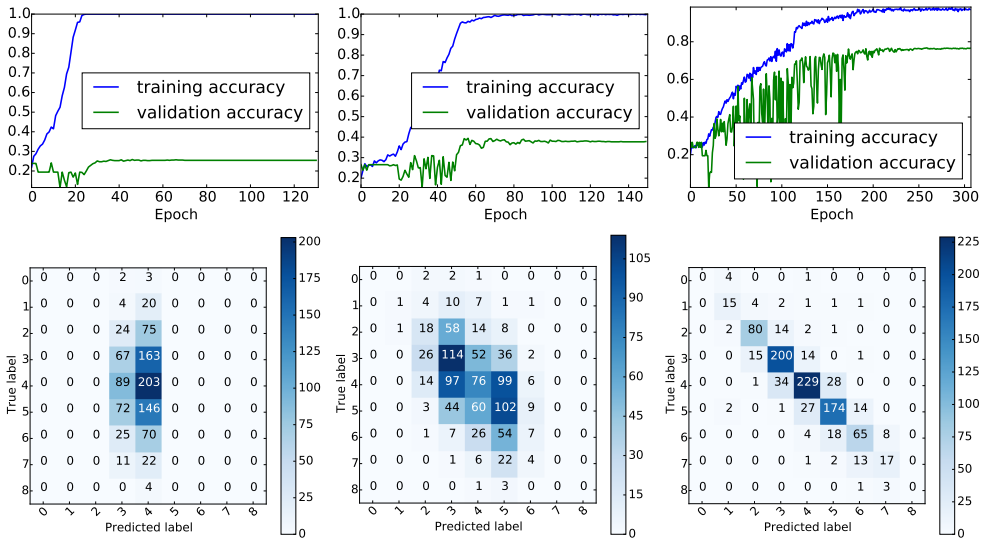


FIGURE 6.7: One leakage protected via uniform RDI: accuracies vs epochs and confusion matrices obtained with our CNN for different DA techniques. From left to right: SH_0, SH_{100}, SH_{500}.

TABLE 6.1: Results of our CNN, for different DA techniques, in presence of an uniform RDI countermeasure protecting. For each technique, 4 values are given: in position a the maximal training accuracy, in position b the maximal validation accuracy, in position c the test accuracy, in position d the value of $N^\star$ (see Sec. 6.6 for definitions).

		SH_0		SH_{100}		SH_{500}	
a	b	100%	25.9%	100%	39.4%	**98.4%**	**76.7%**
c	d	27.0%	>1000	31.8%	101	**78.0%**	**7**

recommended for many applications in literature (see Sec. 6.3), we obtained in most cases better results using the hyperbolic tangent, defined as:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \,. \tag{6.9}$$

We trained our CNN by batches of size 32. In total the network contained $869,341$ trainable weights. The training and validation accuracies achieved after each epoch are depicted in Fig.6.7 together with the confusion matrices that we obtained from the test set. Applying the early-stopping principle recalled in Sec. 6.4.1, we automatically stopped the training after 120 epochs without decrement of the loss function evaluated over the validation set, and kept as final trained model the one that showed the minimal value for the loss function evaluation. Concerning the learning rate (see Sec. 6.4.1), we fixed the beginning one to 0.01 and reduced it multiplying it by a factor of $\sqrt{0.1}$ after 5 epochs without validation loss decrement.

Table 6.1 summarizes the obtained results. For each trained model we can compare the maximal training accuracy achieved during the training with the maximal validation accuracy, defined in Sec. 6.6. This comparison gives an insight about the

risk of overfitting for the training.[6] Case SH_0 corresponds to a training performed without DA technique. When no DA is applied, the overfitting effect is dramatic: the training set is 100%-successfully classified after about 22 epochs, while the test accuracy only achieves 27%. The 27% is around the rate of uniformly distributed bytes showing an Hamming weight of 4.[7] Looking at the corresponding confusion matrix we remark that the CNN training has been biased by the binomial distribution of the training data, and almost always predicts the class 4. This essentially means that no discriminative feature has been learned in this case, which is confirmed by the fact that the trained model leads to an unsuccessful attack ($N^\star > 1,000$). Remarkably, the more artificial shifting is added by the DA, the more the overfitting effect is attenuated; for SH_T with *e.g.* $T = 500$ the training set is never completely learnt and the test accuracy achieves 78%, leading to a guessing entropy of 1 with only $N^\star = 7$ traces.

These results confirm that our CNN model is able to characterize a wide range of points in a way that is robust to RDI.

6.9.2 Two Leaking Operations

Here we study whether our CNN classifier suffers from the presence of multiple leaking operations with the same power consumption pattern. This situation occurs for instance any time the same operation is repeated several successive times over different pieces of data (*e.g.* the SubByte operation for a software AES implementation is often performed by 16 successive look-up table accesses). To start our study we performed the same experiments as in Sec. 6.9.1 over a second traces set, where two look-up table accesses leak, each preceded by a random delay. Some examples of this second traces set are given in the right side of Fig. 6.6, where the two leaking operations being highlighted by red and green ellipses. We trained the same CNN as in Sec. 6.9.1, once to classify the first leakage, and a second time to classify the second leakage, applying SH_{500} as DA technique. Results are given in Table 6.2. They show that even if the CNN transforms spatial (or temporal) information into abstract discriminative features, it still holds an ordering notion: indeed if no ordering notion would have been held, the CNN could no way discriminate the first peak from the second one.

[6] The validation accuracies are estimated over a 700-sized set, while the test accuracies are estimated over $100,000$ traces. Thus the latter estimation is more accurate, and we recall that the test accuracy is to be considered as the final CNN classification performance.

[7] We recall that the Hamming weight of uniformly distributed data follows a binomial law with coefficients $(8, 0.5)$.

TABLE 6.2: Results of our CNN in presence of uniform RDI protecting two leaking operations. See the caption of Table 6.1 for a legend.

		First operation		Second operation	
a	b	95.2%	79.7%	96.8%	81.0%
c	d	76.8%	7	82.5%	6

6.10 Experiments against Artificial Hardware Countermeasures

A classical hardware countermeasure against side-channel attacks consists in introducing instability in the clock. This implies the cumulation of a deforming effect that affects each single acquired clock cycle, and provokes traces misalignment on the adversary side. Indeed, since clock cycles do not have the same duration, they are sampled during the attack by a varying number of time samples. As a consequence, a simple translation of the acquisitions is not sufficient in this case to align with respect to an identified clock cycle. Several realignment techniques are available to manage this kind of deformations, *e.g.* [WWB11]. In this context, our goal is to show that we can get rid of the realignment pre-processing, letting the CNN deep structure take it in charge implicitly.

6.10.1 Performances over Artificial Augmented Clock Jitter

In this section we present the results that we obtained over two datasets named *DS_low_jitter* and *DS_high_jitter*. Each one contains $10,000$ labelled traces, used for the training phase (more precisely, $9,000$ are used for the training, and $1,000$ for the validation), and $100,000$ attack traces. The traces are composed of $1,860$ time samples. The two datasets have been obtained by artificially adding a simulated jitter effect over some synchronised original traces. The original traces were measured on the same Atmega328P microprocessor used in the previous section. We verified that they originally encompass leakage on 34 instructions: 2 *nops*, 16 loads from the NVM and 16 accesses to look-up tables. For our attack experiments, it is assumed that the target is the first look-up table access, *i.e.* the 19th clock cycle. As in the previous section, the target is assumed to take the form $Z = \mathrm{HW}(\mathrm{Sbox}(P \oplus K))$. To simulate the jitter effect we used the technique described in Appendix B, fixing parameters $\texttt{sigma} = 4$, $\texttt{B} = 2$ for the *DS_low_jitter* dataset, and $\texttt{sigma} = 6$, $\texttt{B} = 4$ for the *DS_high_jitter* dataset. In the same Appendix B, some traces of *DS_low_jitter* and *DS_high_jitter* are depicted (respectively in Fig. B.1(a) and in Fig. B.1(b)): the cumulative effect of the jitter is observable by remarking that the desynchronisation raises with time. For both datasets we did not operate any PoI selection, but entered the entire traces into our CNN.

We used the same CNN architecture (6.8) as in previous section. We assisted again to a strong overfitting phenomenon and we successfully reduced it by applying the DA strategy introduced in Sec. 6.8. This time we applied both the *shifting* deformation SH_T with $T^\star = 200$ and $T \in \{0, 20, 40\}$ and the *add-remove* deformation AR_R with $R \in \{0, 100, 200\}$, training the CNN model using the nine combinations $\mathrm{SH}_T\mathrm{AR}_R$. We performed a further experiment with much higher DA parameters, *i.e.* $\mathrm{SH}_{200}\mathrm{AR}_{500}$, to show that the benefits provided by the DA are limited: as expected, too much deformation affects the CNN performances (indeed results obtained with $\mathrm{SH}_{200}\mathrm{AR}_{500}$ will be worse than those obtained with *e.g.* $\mathrm{SH}_{40}\mathrm{AR}_{200}$).

The results we obtained are summarized in Table 6.3. Case $\mathrm{SH}_0\mathrm{AR}_0$ corresponds to a training performed without DA technique, hence serves as a reference suffering from the overfitting phenomenon. It can be observed that as the DA parameters raise, the validation accuracy increases while the training accuracy decreases. This experimentally validates that the DA technique is efficient in reducing overfitting. Remarkably in some cases, for example in the *DS_low_jitter* dataset case with $\mathrm{SH}_{100}\mathrm{AR}_{40}$, the best validation accuracy is higher than the best training accuracy. In Fig. 6.8 the training and validation accuracies achieved in this case epoch by epoch are depicted. It can be noticed that the unusual relation between the training and the validation accuracies does not only concern the maximal values, but is almost kept epoch by epoch. Observing the picture, we can be convinced that, since this fact occurs at many epochs, this is not a consequence of some unlucky inaccurate estimations. To interpret this phenomenon we observe that the training set contains both the original data and the augmented ones (*i.e.* deformed by the DA) while the validation set only contains non-augmented data. The fact that the achieved training accuracy is lower than the validation one, indicates that the CNN does not succeed in learning how to classify the augmented data, but succeeds to extract the features of interest for the classification of the original data. We judge this behaviour positively. Concerning the DA techniques we observe that they are efficient when applied independently and that their combination is still more efficient.

According to our results in Table 6.3, we selected the model issued using the $\mathrm{SH}_{200}\mathrm{AR}_{40}$ technique for the *DS_low_jitter* dataset and the one issued using the $\mathrm{SH}_{200}\mathrm{AR}_{20}$ technique for the *DS_higher_jitter*. In Fig. 6.9 we compare their performances with those of a Gaussian TA combined with a realignment technique. To tune this comparison, several state-of-the-art Gaussian TA have been tested. Since in the experiment the leakage is concentrated in peaks that are easily detected by their relatively high amplitude, we use as realignment technique a simple method that consists in first detecting the peaks above a chosen threshold, then keeping all the samples in a window around these peaks. Then, for the selection of the PoIs,

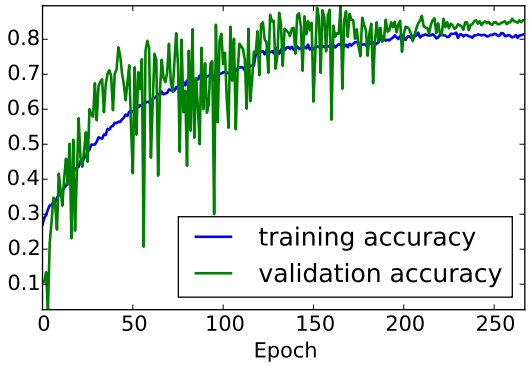


FIGURE 6.8: Training of the CNN model with DA $\mathrm{SH}_{100}\mathrm{AR}_{40}$. The training classification problem becomes harder than the real classification problem, leading validation accuracy constantly higher than the training one.

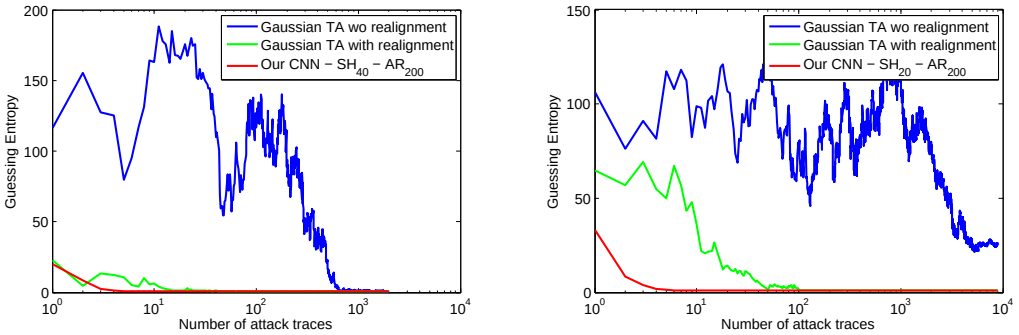


FIGURE 6.9: Comparison between a Gaussian template attack, with and without realignment, and our CNN strategy, over the *DS_low_jitter* (left) and the *DS_high_jitter* (right).

two approaches have been applied: first we selected from 3 to 20 points maximising the estimated instantaneous SNR, secondly we selected sliding windows of 3 to 20 consecutive points covering the region of interest. For the template processing, we tried (1) the classical approach [CRR03] where a mean and a covariance matrix are estimated for each class, (2) the *pooled* covariance matrix strategy proposed in [CK14b] and (3) the stochastic approach proposed in [SLP05]. The results plotted in Fig. 6.9 are the best ones we obtained (via the stochastic approach over some 5-sized windows). Results show that the performances of the CNN approach are much higher than those of the Gaussian templates, both with and without realignment. This confirms the robustness of the CNN approach with respect to the jitter effect: the selection of PoIs and the realignment integrated in the training phase are effective.

TABLE 6.3: Results of our CNN in presence of artificially-generated jitter countermeasure, with different DA techniques. See the caption of Table 6.1 for a legend.

DS_low_jitter									
a	b	SH_0		SH_{20}		SH_{40}		SH_{200}	
c	d								
AR_0		100.0%	68.7%	99.8%	86.1%	98.9%	84.1%		
		57.4%	14	82.5%	6	83.6%	6		
AR_{100}		87.7%	88.2%	82.4%	88.4%	81.9%	89.6%		
		86.0%	6	87.0%	5	87.5%	6		
AR_{200}		83.2%	88.6%	81.4%	86.9%	**80.6%**	**88.9%**		
		86.6%	6	85.7%	6	**87.7%**	**5**		
AR_{500}								85.0%	88.6%
								86.2%	5
DS_high_jitter									
a	b	SH_0		SH_{20}		SH_{40}		SH_{200}	
c	d								
AR_0		100%	45.0%	100%	60.0%	98.5%	67.6%		
		40.6%	35	51.1%	9	62.4%	11		
AR_{100}		90.4%	57.3%	76.6%	73.6%	78.5%	76.4%		
		50.2%	15	72.4%	11	73.5%	9		
AR_{200}		83.1%	67.7%	**82.0%**	**77.1%**	82.6%	77.0%		
		64.0%	11	**75.5%**	**8**	74.4%	8		
AR_{500}								83.6%	73.4%
								68.2%	11

6.11 Experiments against Real-Case Hardware Countermeasures

As a last (but most challenging) experiment we deployed our CNN architecture to attack an AES hardware implementation over a modern secure smartcard (secure implementation on 90nm technology node). On this implementation, the architecture is designed to optimise the area, and the speed performances are not the major concern. The architecture is here minimal, implementing only one hardware instance of the SubByte module. The AES SubByte operation is thus executed serially and one byte is processed per clock cycle. To protect the implementation, several countermeasures are implemented. Among them, a hardware mechanism induces a strong jitter effect which produces an important traces' desynchronisation. The bench is set up to trig the acquisition of the trace on a peak which corresponds to the processing of the first byte. Consequently, the set of traces is aligned according to the processing of the first byte while the other bytes leakages are completely misaligned. To illustrate the effect of this misalignment, the SNR characterising the (aligned) first byte and the (misaligned) second byte are computed (according to (2.1)) using a set of $150,000$ traces labelled by the value of the SubByte output (256 labels). These

SNRs are depicted in the top part of Fig. 6.10. The SNR of the first byte (in green) detects a quite high leakage, while the SNR of the second byte (in blue) is nullified. A zoom of the SNR of the second peak is proposed in the bottom part of Fig. 6.10. In order to confirm that the very low SNR corresponding to the second byte is only due to the desynchronisation, the patterns of the traces corresponding to the second byte have been resynchronised using a peak-detection-based algorithm, quite similar to the one applied for the experiments of Sec. 6.10.1. Then the SNR has been computed onto these new aligned traces and has been plot in red in the top-left part of Fig. 6.10; this SNR is very similar to that of the first byte. This clearly shows that (1) the leakage information is contained into the trace but is efficiently hidden by the jitter-based countermeasure, and that (2) the realignment technique we applied in this context is effective.

We applied the CNN approach onto the rough set of traces (without any alignement). First, a $2,500$-long window of the trace has been selected to input CNN. The window, identified by the vertical cursors in the bottom part of Fig. 6.10, has been selected to ensure that the pattern corresponding to the leakage of the second byte is inside the selection. At this step, it is important to notice that such a selection is not at all as meticulous as the selection of PoIs required by a classical TA approach. The training phase has been performed using $98,000$ labelled traces; $1,000$ further traces have been used for the validation set. We performed the training phase over a desktop computer equipped with an Intel Xeon E5440 @2,83GHz processor, 24Gb of RAM and a GeForce GTS 450 GPU. Without data augmentation each epoch took about 200s.[8] The training stopped after 25 epochs. Considering that in this case we applied an early-stopping strategy that stopped training after 20 epochs without validation loss decrement, it means that the final trainable weights are obtained after 5 epochs (in about 15 minutes). The results that we obtained are summarized in Table 6.4. They prove not only that our CNN is robust to the misalignment caused by the jitter but also that the DA technique is effective in raising its efficiency. A comparison between the CNN performances and the best results we obtained over the same dataset applying the realignment-TA strategy, is proposed in Fig. 6.11. Beyond the fact that the CNN approach slightly outperforms the realignment-TA one, and considering that both case-results shown here are surely non-optimal, what is remarkable is that the CNN approach is potentially suitable even in cases where realignment methods are impracticable or not satisfying. It is of particular interest in cases where sensitive information does not lie in proximity of peaks or of easily detectable patterns, since many resynchronisation techniques are based on pattern or peak detection. If the resynchronisation fails, the TA approach falls out of service,

[8] raising to about $2,000$ seconds when $SH_{20}DA_{200}$ data augmentation is performed (data are augmented online during training)

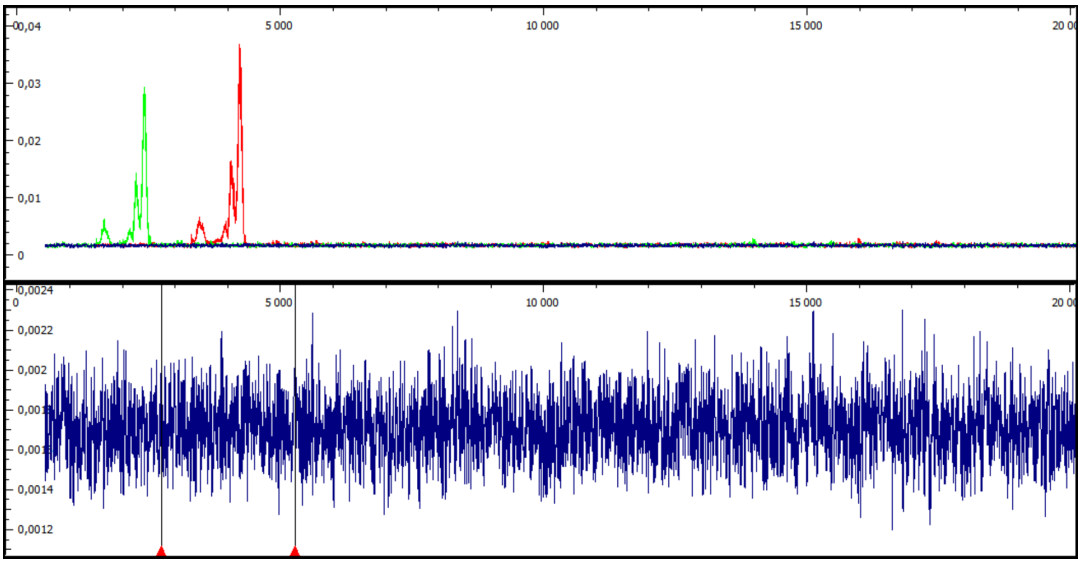


FIGURE 6.10: AES hardware implementation protected by jitter-based misalignment. In green the SNR for the first byte; in blue the SNR for the second byte; in red the SNR for the second byte after a trace realignment.

		SH_0AR_0		$SH_{10}AR_{100}$		$SH_{20}AR_{200}$	
a	b	35.0%	1.1%	12.5%	1.5%	**10.4%**	**2.2%**
c	d	1.2%	137	1.3%	89	**1.8%**	**54**

TABLE 6.4: Results of our CNN over the modern smart card with jitter.

while the CNN one remains a further weapon in the hands of an attacker.

6.12 Conclusion

In this chapter, we have proposed an end-to-end profiling attack approach, based on the CNNs. We claimed that such a strategy would keep effective even in presence of trace misalignment, and we successfully verified our claim by performing CNN-based attacks against different kinds of misaligned data. This property represents a great practical advantage compared to the state-of-the-art Template Attacks, that require a meticulous trace realignment in order to be efficient. Our strategy based over CNNs differs from classical TA for mainly two points. First, it makes use of a discriminative model, instead of a generative one. Second it takes in charge into a unique training phase all eventual preprocessing phases necessary for the successfulness of a TA. Indeed, beyond the trace realignment, that is not necessary for the CNN approach, it represents as well a solution to the problem of the selection

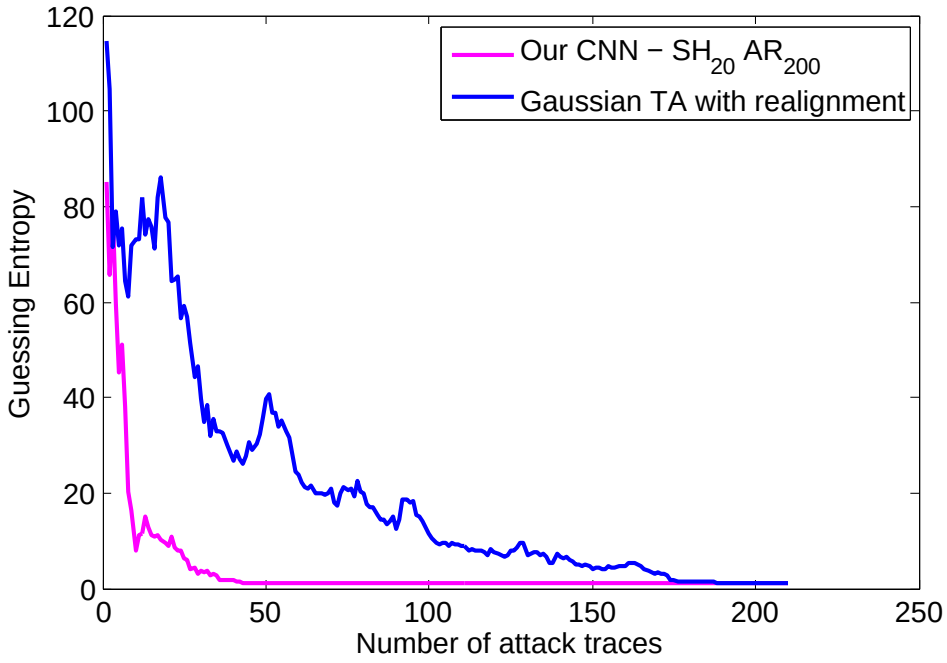


FIGURE 6.11: Comparison between a Gaussian template attack with realignment, and our CNN strategy, over the modern smart card with jitter.

of points of interest issue: CNNs efficiently manage high-dimensional data, allowing the attacker to simply select large windows. In this sense, the experiments described in Sec. 6.11 are very representative: our CNN retrieves information from a large window of points instead of an almost null instantaneous SNR. To guarantee the robustness to trace misalignment, we used a quite complex architecture for our CNN, and we clearly identified the risk of overfitting phenomenon. To deal with this classical issue in machine learning, we proposed two Data Augmentation techniques adapted to misaligned side-channel traces. All the experimental results we obtained have proved that they provide a great benefit to the CNN strategy. Attacks proposed in this chapter are performed against non-masked implementation. Nevertheless, since NNs are in general non-linear models, they naturally well-fit also the higher-order attack context, as discussed in [MPP16] and [Pro+18].

Chapter 7

Conclusions and Perspectives

7.1 Summary

7.2 Tracks for Future Works

7.2.1 Towards a Side-Channel Deep Learning Community

ASCAD
definition of a DPA-specific machine learning task and proposition for specific metrics (*e.g.* loss function, evaluation metrics...)

7.2.2 Bayesian Inference

and Bayesian Deep Learning...see https://alexgkendall.com/computer_vision/bayesian_deep_learning_for_safe_ai/

7.2.3 Strengthen Embedded Security against Powerful Increasing Machine Learning Attackers

from a successful attack understand which part of execution most contribute for the success...

Appendix A

Cross-Validation

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

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

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

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

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

Appendix B

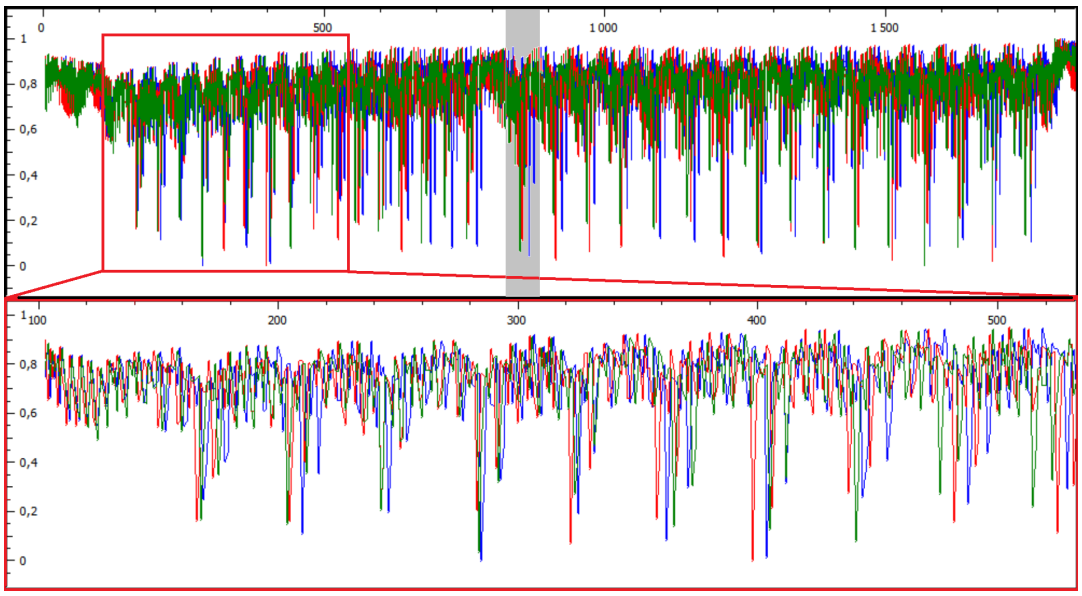
Artificially Simulate Jitter

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

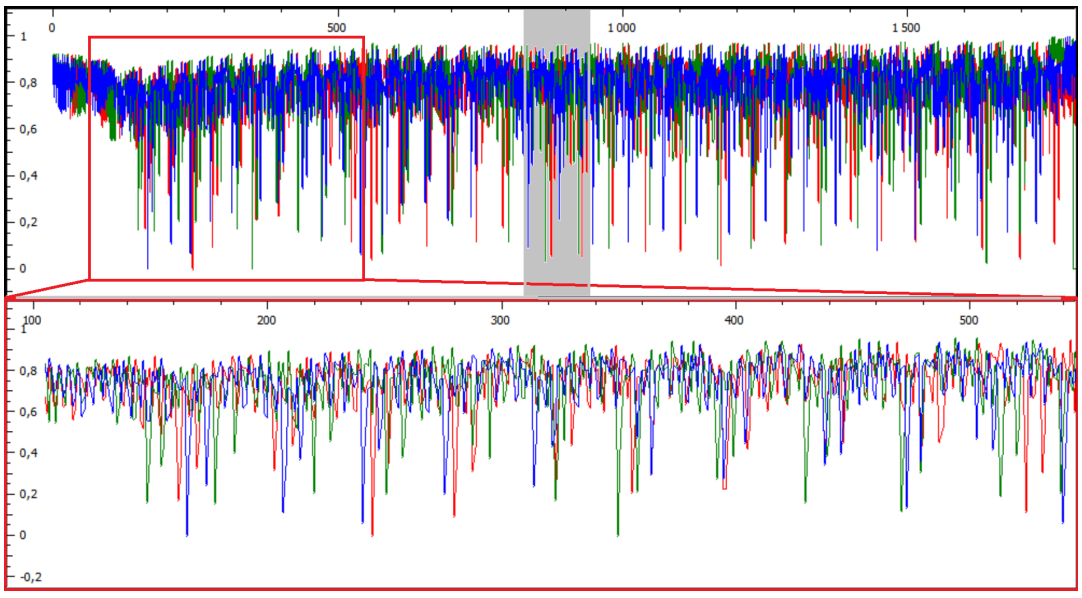
```
def enlarge_reduce_window(window,sigma,B):
    Npts = window.shape[0]
    new_window = np.copy(window)
    deltaPts = int(np.floor(np.random.randn(1)[0]*sigma))
    if (deltaPts >= 0):
        deltaPts = min(Npts*(B-1),deltaPts)
        for i  in range(deltaPts):
            curr_size = new_window.shape[0]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

```
        deltaPts = max(-Npts*(1-1/B), deltaPts)
        for i in range(-deltaPts):
            curr_size = new_window.shape[0]
            pos = int(np.floor(np.random.rand(1)*curr_size))
            new_window = np.delete(new_window, pos)
    return new_window
```

This deformation is applied to each clock pattern independently. We remark that is implies that, for example, The 19th clock cycle of a deformed acquisition suffers from the cumulation of the previous 18 deformations. For the sake of visualizing the effect of such a jitter simulation, in Fig. B.1 we depict some traces of *DS_low_jitter* B.1(a) and of the *DS_high_jitter* B.1(b) datasets, used for experiments in Sec .6.10. 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.



(a)



(b)

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

Appendix C

Kernel PCA construction

Suppose that we want to perform PCA in the image space of a function Φ that is associated to a given kernel function K. The kernel version for PCA has been presented in [SSM98]; as we said in Chapter 5, the important step consists in expressing the operations needed for the PCA procedure in terms of the dot products between the mapped data.

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

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

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

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

We remark that such an eigenvector satisfies

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

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

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

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

[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.

where the step (C.4) makes use of the associativity of the matrix product and the commutativity of the scalar-matrix product. Eq. (C.6) tells us that each eigenvector $\vec{\alpha}^{\Phi}$ is expressible as a linear combination of the data mapped into the feature space $(\Phi(\vec{x}_i)_{i=1,\dots,N_p}$, or equivalently each eigenvector $\vec{\alpha}^{\Phi}$ lies in the span of $(\Phi(\vec{x}_i)_{i=1,\dots,N_p})$. This observation authorizes to substitute to the problem (C.2), the following equivalent system:

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

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

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

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

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

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

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

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

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

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

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

which equals the following equation:

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

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

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

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

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

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

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

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

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

C.1 Kernel class-oriented PCA

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Bibliography

[Abi] *8.8 Billion Smart Cards Shipped in 2014 Driven by Growth in the Banking and SIM Card Markets*. https://www.abiresearch.com/press/88-billion-smart-cards-shipped-in-2014-driven-by-g/. Accessed: 2018-07-02 (cit. on p. 8).

[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. 57, 58, 64).

[BA00] Gaston Baudat and Fatiha Anouar. "Generalized discriminant analysis using a kernel approach". In: *Neural computation* 12.10 (2000), pp. 2385–2404 (cit. on p. 87).

[Bar+15] Gilles Barthe et al. "Verified Proofs of Higher-Order Masking". In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*. 2015, pp. 457–485. DOI: 10.1007/978-3-662-46800-5_18. URL: https://doi.org/10.1007/978-3-662-46800-5_18 (cit. on p. 18).

[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 pp. 32, 33).

[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 pp. 32, 33).

[BDP10] Martin Bär, Hermann Drexler, and Jürgen Pulkus. "Improved template attacks". In: *COSADE2010* (2010) (cit. on p. 42).

[Bel+15] Sonia Belaïd et al. "Improved Side-Channel Analysis of Finite-Field Multiplication". In: *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16,*

2015, Proceedings. 2015, pp. 395–415. DOI: 10.1007/978-3-662-48324-4_20 (cit. on p. 118).

[BG05] Yoshua Bengio and Yves Grandvalet. "Bias in estimating the variance of K-fold cross-validation". In: *Statistical modeling and analysis for complex data problems*. Springer, 2005, pp. 75–95 (cit. on p. 131).

[BHK97] Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. 1997 (cit. on pp. 59, 71).

[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. 57, 58, 65).

[Bis06] Christopher M.. Bishop. *Pattern recognition and machine learning*. Springer, 2006 (cit. on pp. 38, 39, 46, 60, 97, 108, 115).

[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. 53).

[Bog07] Andrey Bogdanov. "Improved side-channel collision attacks on AES". In: *International Workshop on Selected Areas in Cryptography*. Springer. 2007, pp. 84–95 (cit. on p. 31).

[Bog08] Andrey Bogdanov. "Multiple-differential side-channel collision attacks on AES". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2008, pp. 30–44 (cit. on p. 31).

[Bre+96] Leo Breiman et al. "Heuristics of instability and stabilization in model selection". In: *The annals of statistics* 24.6 (1996), pp. 2350–2383 (cit. on p. 131).

[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. 57, 68).

[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. 85).

[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 p. 83).

[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. 57, 66, 78).

[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. 19, 81).

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. "Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing". In: *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. Lecture Notes in Computer Science. Springer, 2017, pp. 45–68. ISBN: 978-3-319-66786-7. DOI: 10.1007/978-3-319-66787-4_3. URL: https://doi.org/10.1007/978-3-319-66787-4_3 (cit. on pp. 20, 101).

[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. 57).

[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. 83, 102).

[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. 59, 71).

[Cho+15] François Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015 (cit. on p. 118).

[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 pp. 101, 117).

[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 pp. 101, 117).

[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. 57).

[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. 40, 57, 58, 64, 123).

[CK14c] Omar Choudary and Markus G Kuhn. "Template attacks on different devices". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2014, pp. 179–198 (cit. on p. 78).

[CK18] Marios O Choudary and Markus G Kuhn. "Efficient, Portable Template Attacks". In: *IEEE Transactions on Information Forensics and Security* 13.2 (2018), pp. 490–501 (cit. on p. 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. 92).

[Cla+10] Christophe Clavier et al. "Horizontal correlation analysis on exponentiation". In: *International Conference on Information and Communications Security*. Springer. 2010, pp. 46–61 (cit. on pp. 31, 32).

[Cla+12] Christophe Clavier et al. "ROSETTA for single trace analysis". In: *Progress in Cryptology-INDOCRYPT 2012: 12th International Conference on Cryptology in India, Chennai, India, December 11-14, 2011, Proceedings13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012, Proceedings*. Vol. 7668. Springer. 2012, p. 140 (cit. on p. 30).

[CMW14] Christophe Clavier, Damien Marion, and Antoine Wurcker. "Simple power analysis on AES key expansion revisited". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2014, pp. 279–297 (cit. on p. 30).

[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. 32, 37, 38, 39, 41, 123).

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297 (cit. on p. 53).

[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. 84).

[Dur+12] François Durvaux et al. "Efficient removal of random delays from embedded software implementations using hidden markov models". In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2012, pp. 123–140 (cit. on pp. 102, 117).

[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. 58, 84).

[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. 57).

[Fel08] Willliam Feller. *An introduction to probability theory and its applications*. Vol. 2. John Wiley & Sons, 2008 (cit. on p. 24).

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, 2001 (cit. on p. 131).

[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. 58).

[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. 68, 70).

[FV03] Pierre-Alain Fouque and Frédéric Valette. "The doubling attack–why upwards is better than downwards". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2003, pp. 269–280 (cit. on p. 31).

[GBC16a] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 106, 107).

[GBC16b] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN: 978-0-262-03561-3. URL: http://www.deeplearningbook.org/ (cit. on p. 107).

[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. 9).

[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. 9).

[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. 41, 42).

[GMGW98] T. Guhr, A. Müller-Groeling, and H. A. Weidenmüller. "Random-matrix theories in quantum physics: common concepts". In: *Physics Reports* 299.4 (1998), pp. 189–425 (cit. on p. 65).

[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. 9).

[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. 9).

[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. 9).

[Hin+12] Geoffrey E. Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580 (2012). URL: http://arxiv.org/abs/1207.0580 (cit. on p. 115).

[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. 53).

[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. 59, 72).

[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. 53).

[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. 101).

[IS15] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). URL: http://arxiv.org/abs/1502.03167 (cit. on p. 118).

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. "Private circuits: Securing hardware against probing attacks". In: *Annual International Cryptology Conference*. Springer. 2003, pp. 463–481 (cit. on p. 18).

[Kar+09] Peter Karsmakers et al. *Side channel attacks on cryptographic devices as a classification problem*. Tech. rep. COSIC technical report, 2009 (cit. on p. 57).

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. "Differential power analysis". In: *Annual International Cryptology Conference*. Springer. 1999, pp. 388–397 (cit. on pp. 9, 32, 36).

[Koc+11] Paul Kocher et al. "Introduction to differential power analysis". In: *Journal of Cryptographic Engineering* 1.1 (2011), pp. 5–27 (cit. on pp. 30, 31).

[Koc96] Paul C Kocher. "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems". In: *Annual International Cryptology Conference*. Springer. 1996, pp. 104–113 (cit. on pp. 9, 27).

[KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.* 2012, pp. 1106–1114 (cit. on pp. 101, 115).

[LB+95] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995 (cit. on p. 111).

[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. 53).

[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. 53).

[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. 71).

[LeC+12] Yann A. LeCun et al. "Efficient BackProp". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3. URL: http://dx.doi.org/10.1007/978-3-642-35289-8_3 (cit. on p. 107).

[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. 53).

[LH05] Yann LeCun and Fu Jie Huang. "Loss Functions for Discriminative Training of Energy-Based Models". In: *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*. 2005. URL: http://www.gatsby.ucl.ac.uk/aistats/fullpapers/207.pdf (cit. on p. 107).

[Lib13] Joint Interpretation Library. *Application of Attack Potential to SmartSmart, Version 2.9*. Tech. rep. Common Criteria, 2013 (cit. on p. 15).

[Lio+14] Rokach Lior et al. *Data mining with decision trees: theory and applications*. Vol. 81. World scientific, 2014 (cit. on p. 53).

[LMV04] Hervé Ledig, Frédéric Muller, and Frédéric Valette. "Enhancing collision attacks". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2004, pp. 176–190 (cit. on p. 31).

[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. 83).

[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. 41).

[LRP07] Kerstin Lemke-Rust and Christof Paar. *Gaussian mixture models for higher-order side channel analysis*. Springer, 2007 (cit. on p. 83).

[LWH90] Kevin J Lang, Alex H Waibel, and Geoffrey E Hinton. "A time-delay neural network architecture for isolated word recognition". In: *Neural networks* 3.1 (1990), pp. 23–43 (cit. on p. 101).

[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. 92).

[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. 30).

[Man04] Stefan Mangard. "Hardware countermeasures against DPA–a statistical analysis of their effectiveness". In: *Topics in Cryptology–CT-RSA 2004*. Springer, 2004, pp. 222–235 (cit. on p. 102).

[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. 59, 65, 66).

[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. 54).

[MHK10] David A. McAllester, Tamir Hazan, and Joseph Keshet. "Direct Loss Minimization for Structured Prediction". In: *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.* 2010, pp. 1594–1602. URL: http://papers.nips.cc/paper/4069-direct-loss-minimization-for-structured-prediction (cit. on p. 109).

[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. 54).

[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. 101).

[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. 54).

[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. 101).

[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. 101).

[MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*. Vol. 31. Springer Science & Business Media, 2008 (cit. on p. 41).

[Mor74] Roland Moreno. *Methods of data storage and data storage systems*. US3971916A. 1974 (cit. on p. 8).

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. "Breaking Cryptographic Implementations Using Deep Learning Techniques". In: *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer. 2016, pp. 3–26 (cit. on pp. 54, 127).

[MZ13] Zdenek Martinasek and Vaclav Zeman. "Innovative method of the power analysis". In: *Radioengineering* 22.2 (2013), pp. 586–594 (cit. on p. 54).

[Nag+07] Sei Nagashima et al. "DPA using phase-based waveform matching against random-delay countermeasure". In: *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE. 2007, pp. 1807–1810 (cit. on pp. 102, 117).

[NH10] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814 (cit. on pp. 106, 118).

[NIS] FIPS PUB NIST. *197," Advanced Encryption Standard (AES)," November 2001* (cit. on pp. 4, 5, 6, 7).

[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. 73, 89).

[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. 81).

[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*. ACM. 2013, p. 7 (cit. on p. 33).

[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. 33).

[Par] TELECOM ParisTech. "DPA Contest 4". In: (). http://www.DPAcontest.org/v4/. URL: http://www.DPAcontest.org/v4/ (cit. on p. 65).

[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. 39).

[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. 18).

[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. 83).

[Pre12] Lutz Prechelt. "Early Stopping — But When?" In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67. ISBN: 978-3-642-35289-8. DOI: 10.1007/

978-3-642-35289-8_5. URL: http://dx.doi.org/10.1007/978-3-642-35289-8_5 (cit. on p. 107).

[Pro+18] Emmanuel Prouff et al. *Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database*. Cryptology ePrint Archive, Report 2018/053. https://eprint.iacr.org/2018/053. 2018 (cit. on p. 127).

[Pu+17] Sihang Pu et al. "Trace Augmentation: What Can Be Done Even Before Preprocessing in a Profiled SCA?" In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2017, pp. 232–247 (cit. on p. 116).

[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. 9).

[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. 83).

[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. 53).

[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 pp. 36, 41).

[RS09] Mathieu Renauld and François-Xavier Standaert. "Algebraic Side-Channel Attacks." In: *Inscrypt* 6151 (2009), pp. 393–410 (cit. on p. 33).

[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. 33).

[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. 57, 68, 70).

[Sch+04] Kai Schramm et al. "A collision-attack on AES". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2004, pp. 163–175 (cit. on p. 31).

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. "A stochastic model for differential side channel cryptanalysis". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2005, pp. 30–46 (cit. on p. 123).

[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. 87, 90, 91).

[Son+15] Yang Song et al. "Direct Loss Minimization for Training Deep Neural Nets". In: *CoRR* abs/1511.06411 (2015). URL: http://arxiv.org/abs/1511.06411 (cit. on p. 109).

[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. 58, 65).

[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. 87, 137).

[SSP+03] Patrice Y Simard, David Steinkraus, John C Platt, et al. "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis." In: *ICDAR*. Vol. 3. Citeseer. 2003, pp. 958–962 (cit. on pp. 50, 115).

[SWP03] Kai Schramm, Thomas Wollinger, and Christof Paar. "A new class of collision attacks and its application to DES". In: *International Workshop on Fast Software Encryption*. Springer. 2003, pp. 206–222 (cit. on p. 31).

[TB07] Michael Tunstall and Olivier Benoit. "Efficient use of random delays in embedded software". In: *IFIP International Workshop on Information Security Theory and Practices*. Springer. 2007, pp. 27–38 (cit. on p. 118).

[TM97] Mitchell T. M. *Machine Learning*. McGraw-Hill, New York, 1997 (cit. on p. 43).

[Ugo77] Michel Ugon. *Portable data carrier including a microprocessor*. US4211919A. 1977 (cit. on p. 8).

[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. 101).

[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. 33).

[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. 53).

[WW98] Jason Weston and Chris Watkins. *Multi-class support vector machines*. Tech. rep. Citeseer, 1998 (cit. on p. 53).

[WWB11] Jasper GJ van Woudenberg, Marc F Witteman, and Bram Bakker. "Improving differential power analysis by elastic alignment". In: *Cryptographersí Track at the RSA Conference*. Springer. 2011, pp. 104–119 (cit. on pp. 102, 121).

[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. 59, 71).

[Zho+17] Xinping Zhou et al. "A Novel Use of Kernel Discriminant Analysis as a Higher-Order Side-Channel Distinguisher". In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2017, pp. 70–87 (cit. on pp. 81, 97).