Evaluating the security of cryptographic systems against electromagnetic attacks

Ricardo Jorge do Rosário Maçãs

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Dr. Ricardo Jorge Fernandes Chaves
Prof. Dr. Gonçalo Nuno Gomes Tavares

Examination Committee

Chairperson: Prof. Dr. José Eduardo Charters Ribeiro da Cunha Sanguino
Supervisor: Prof. Dr. Ricardo Jorge Fernandes Chaves
Member of the Committee: Prof. Dr. Pedro Miguel dos Santos Alves Madeira Adão

November 2017

in memoriam

Júlia Ribeiro Maçãs (1945-2016)

Filipe Alexandre Maçãs Luís (1995-2016)

Adelino Júlio do Rosário (1930-2017)

> “From the point of view of the cryptanalyst, a secrecy system is almost identical with a noisy communication system.
>
> The message (transmitted signal) is operated on by a statistical element, the enciphering system, with its statistically chosen key. The result of this operation is the cryptogram (analogous to the perturbed signal) which is available for analysis. (...)
>
> One can imagine, for example an intercepted communication between Martian invaders, the a priori probabilities would probably be so uncertain as to be devoid of significance.”
>
> *Claude Elwood Shannon, ‘Communication theory of secrecy systems’*

Acknowledgments

I would like to thank my thesis supervisors, Prof. Ricardo Chaves and Prof. Gonçalo Tavares, for their incredible support throughout these last few months. I have learned immensely with your help and I would like to thank you for enduring my relentless questioning and for your enthusiasm in this adventure. It's been a blast.

To INESC-ID for the warm welcome, especially to my lab colleagues, Ruben Afonso, who was always ready to help in any occasion, and João Lopes, who mentored me through the beginning of this journey.

To Prof. Mário Figueiredo, for always being up to a challenge when I had an interesting question. Thank you for your time and valuable insight.

I'd also like to thank my friends at Técnico Lisboa, Paul Schydlo, an imaginative and always inquisitive soul that has always stood by me, Francisco Silva, a tinkerer and a true engineer at heart, Hugo Loureiro, an intelligent and fierce spirit that never seizes to amaze me, Pedro Ferreira, truly one of the most open-minded and creative people I've met, Henrique Cavaleiro, a fearless mind ready to face the world, and Miguel Gameiro, the cleverest guy I have known in my life. You have been my second family.

Finally, and most importantly, to my family, who stood by me all and supported me through this journey. To my father, Henrique Maçãs, a decisively electrotechnical engineer in spirit, and to my mother, Ana Paula Maçãs, who never left my side and never stopped caring for me. And to my sister, Bianca Maçãs, without doubt the most precious soul to me. I wouldn't be here without all of you.

To all of you, *Obrigado.*

Resumo

A segurança de dispositivos electrónicos é essencial para a nossa sociedade da informação. A maioria dos dispositivos que existem já implementa algoritmos de segurança padronizados, mas sofre de problemas de implementação na sua concretização física. Os *ataques de canal lateral*[1], conhecidos também por *ataques por canal escondido*[2] ou *ataques de canal auxiliar*, permitem a um atacante compromenter um dispositivo através das emanações que decorrem da sua operação, como por exemplo, a radiação electromagnética emitida pelo mesmo. Em particular, os ataques por canal electromagnético podem ser executados sem contacto com o dispositivo, e por isso são particularmente difíceis de detectar e prevenir.

Existem vários tipos de ataques que tiram partido das emanações electromagnéticas, nomeadamente, ataques de correlação e ataques de modelação (*template*). Naturalmente, a execução dos ataques desencadeia-se numa certa configuração experimental, incluindo técnicas de processamento de sinal que preparam os sinais para ataque. No entanto, os resultados existentes na literatura científica focam-se essencialmente no desempenho de cada tipo de ataque, descurando o efeito da configuração experimental e das técnicas de processamento de sinal empregues.

O propósito desta dissertação é realizar uma avaliação rigorosa de ataques electromagnéticos, através de uma comparação empírica das diferentes metodologias de ataque, de forma a permitir a aquisição de conhecimento técnico para a sua realização. Primeiro, é definida a metodologia de análise, isto é, uma estrutura para análise de ataques que permita obter resultados consistentes e reproduzíveis, que depois será usada para comparar as diferentes técnicas. Finalmente, são realizadas avaliações dos efeitos das diferentes configurações experimentais e das técnicas de processamento de sinal empregues para melhorar o desempenho de ataque.

Palavras-chave: criptografia, processamento de sinal, ataques de canal lateral, ataques electromagnéticos, smartcard

[1] do inglês, *side-channel attacks*
[2] do francês, *attaques par canaux cachés*

Abstract

The security of electronic devices is paramount to our information society. The majority of the devices already implement standardized secure algorithms, but suffer from implementation issues in their physical realization. Side-channel attacks enable a potential attacker to stealthily compromise the devices in question by exploiting their emanations, such as the electromagnetic radiation resulting from the operation of the device. In particular, electromagnetic side-channel attacks can be carried out without physically tampering the device, and are thus particularly hard to detect and prevent.

There are several known types of attacks that can exploit the electromagnetic emanations of devices, namely, correlation attacks and template attacks. When these attacks are tested, they are so in some experimental setting, often paired with signal processing techniques that prepare the signals for the attack. However, the results of existing literature often focus on the performance of the type of attack, neglecting to evaluate the effect of the experimental setup and signal processing techniques.

The purpose of this thesis is to conduct a thorough evaluation of the electromagnetic attacks, comparing different attack methodologies empirically, in order to enable the acquisition of know-how for carrying out these attacks. First, the methodology of analysis is defined, i.e., a framework for analyzing every attack that ensures consistent and repeatable results, which can then be used to compare the different techniques. Finally, evaluations of the effects of different experimental setups are conducted, and signal processing techniques are implemented to improve attack performance.

Keywords: cryptography, signal processing, side-channel attacks, electromagnetic attacks, smartcard

Contents

List of Tables

List of Figures

Nomenclature

AES	Advanced Encryption Standard	(block cipher)
APDU	Application Protocol Data Units	(communication unit for smartcards)
ASIC	Application-Specific Integrated Circuit	(nonprogrammable integrated circuit)
CMOS	Complementary Metal-Oxide Semiconductor	(integrated circuits technology)
DC	Direct Current	
DPA	Differential Power Attack	(type of attack)
DUA	Device under Attack	
EM	Electromagnetic	(as in electromagnetic radiation)
FPGA	Field-Programmable Gate Array	(programmable integrated circuit)
IEC	International Electrotechnical Commission	(organization)
I/O	Input/Output	(communication interface)
ISO	International Standards Organization	(organization)
JTAG	Joint Test Action Group	(a standard programming port)
NAND	Negative-AND	(logical gate)
PC	Personal Computer	
SMA	SubMiniature A	(coaxial connector)
SNR	Signal to Noise Ratio	
SPA	Simple Power Attack	(type of attack)

Chapter 1

Introduction

The information society is ever more pervasive in our lives: it permeates into all aspects of society, replacing older technology with connected, intelligent devices. These devices enable all sorts of applications that once were not possible, such as unlocking doors remotely, monitoring sensors and putting their data into context, or controlling implanted devices without having to physically extract them.

As a result, these devices have particular security requirements to ensure that they cannot be tampered with to operate outside their intended use. However, the evolution of technology also means that new methods to compromise them have been discovered which are able to defeat their security. Thus, it is of the utmost importance to understand the attacks that can be carried out in order to be able to design systems that can defeat them.

Not only mathematical attacks need to be considered. Even with an hypothetical perfectly secure algorithm, the operation of electronic devices naturally gives rise to emanations, such as power fluctuations and electromagnetic radiation, which are related to the computation performed by the device. The study of these leakages can reveal information about the operations being executed in the device, which can be exploited to an attacker's advantage to compromise the device's security. Historically, device vendors have approached security as an abstracted layer implemented in software for ensuring the security of their devices, leaving them vulnerable to physical exploitation.

In particular, electromagnetic (EM) emanations have been shown to be highly effective, since they can be captured using common equipment and with minimal interaction with the device, allowing for stealthy attacks which are hard to detect. This grants them a special interest in the cryptanalytic community. The focus of this work is precisely in this type of attacks — the Electromagnetic Side Channel Attacks —, the related techniques and how they can be carried out.

1.1 Objectives

The aim of this work is to contribute to the study of electromagnetic side-channel attacks, by analyzing the existing knowledge and by designing and implementing signal processing techniques to improve them. For this, the execution and evaluation of successful electromagnetic side-channel attacks is pro-

posed, to validate the existing techniques and enable the acquisition of the know-how needed to execute the attacks. Particularly, the use of signal processing techniques applied to these attacks will be evaluated to assess their performance, efficiency and efficacy. These techniques improve the attacks by dealing with the particularities of real traces.

1.2 Main contributions

To accomplish these objectives, the following contributions are proposed:

— Acquisition of the know-how needed to perform the attacks in real cryptographic devices;

— Evaluation of the hardware and software requirements for carrying out a successful side-channel attack with adequate signal processing;

— Evaluation of the side-channel techniques required to perform an electromagnetic side-channel attack to a real cryptographic device;

— Evaluation of the best signal processing techniques for improving electromagnetic side-channel attacks.

1.3 Outline of this thesis

This dissertation will start by addressing the background and foundations of side-channel attacks in Chapter 2, starting with a brief overview of the basic concepts in cryptography, in Section 2.1, and explaining the phenomena that give rise to leakage, in Section 2.2. A brief introduction to cryptographic algorithms is given in Section 2.3.

The state of the art is presented in Chapter 3, describing the latest techniques proposed to perform side-channel attacks. Of particular interest are the correlation attacks, in Section 3.3, and template attacks, presented in Section 3.4.

Chapter 4 presents the methodology of analysis, which sets the criteria used to evaluate and benchmark the performance of the attacks in a controlled and repeatable manner.

The results of this dissertation are divided in two chapters. In Chapter 5, the experimental setup and results are presented for power attacks. The know-how acquired in these attacks is then utilized to successfully carry out the electromagnetic attacks, detailed in Chapter 6, which includes the revised setup and the final results.

Finally, a brief conclusion is presented in Chapter 7 with the key takeaways from the dissertation, followed by Chapter 8 which presents the future work in this area of study.

Chapter 2

Background

Cryptographic devices were first revealed to be vulnerable to key recovery attacks via electromagnetic emanations in Kocher et al.'s [1] paper. At the time, it started a whole new class of attacks, building on the knowledge of modern security that began to develop during World War II with the notable work of Claude Shannon [2] in the field of information theory and what was then the beginning of modern security knowledge. The next sections present a discussion of the concepts in security and cryptology followed by an overview of the physical operation of the devices and how it ties to their electromagnetic emanations. Subsequently, an explanation on the details of modern ciphers is also given. Finally, a study of the several techniques now known as side-channel attacks, which bring the physical and cryptological domains together in order to compromise the device, is presented.

2.1 Concepts in cryptology

First, a careful discussion of the principles and main concepts of security is warranted. Modern systems have several security requirements, some of the most important [3] being *confidentiality* — where messages are ensured to be kept secret between authorized parties, *integrity* — where messages are ensured to be intact and free of tampering, *authentication* — where the identity of an entity is ensured to be correct, and *non-repudiation* — where the facts are ensured to be undeniable, i.e., a mechanism that ensures that if a commitment or action is made, its author cannot repudiate it. These requirements allow us to implement sensitive systems such as, for example, digital payments, since these need a way to verify the user and the payment details without revealing them in public and without allowing the user to dispute the fact that a payment was made. To implement these systems, a suite of algorithms is used that meet the security requirements needed — a cryptosystem.

Definition 1. *A* cryptosystem *is a scheme that algorithmically implements one or more security requirements.*

These cryptosystems are implemented physically, either on general-purpose hardware or on dedicated hardware. Maintaining the example of digital payments, they are often made with a smartcard

issued by the credit card company. This smartcard implements a cryptosystem as a dedicated cryptographic device which allows the bank to securely accept payments made all over the world.

Definition 2. *A* cryptographic device *is a physical implementation of a cryptosystem.*

For confidentiality, the goal is to keep the intended message secret from third-parties in a communication. For example, an user named Alice wants to send a confidential message to Bob (see Figure 2.1). Alice knows that the channel used to communicate is public, that is, other people can read the messages she is sending. What she sends must not only be unrecognizable, but she must also be sure that others cannot know the meaning of the message while guaranteeing the destination can read the original.

To achieve this, both communicating parties must have some algorithm and a secret. One refers to the algorithm as a *cipher*, having some secret *key*, taking as an input some *message* (or interchangeably, *cleartext*) and returning the *ciphertext* (also referred to as *cryptogram*).

Alice will take her message and process it using a cipher and a key. The cipher will do some computation in order to produce an output that hides the intended message but that can only be reversed by those with the key. This process is called *encryption*.

Provided that Bob has the same key, when receiving the ciphertext he can recover the cleartext by using the cipher. Reverting the encryption process is denominated *decryption*. The exact process depends on the type of cipher, but it is akin to running the same algorithm as in encryption but in reverse order.

This kind of encryption is called a *symmetric encryption* scheme, since it uses a single key for encryption and decryption. There are other schemes, denominated *asymmetric encryption* schemes, where two keys exist, in which case Alice would have the key used for encryption while Bob would own a corresponding decryption key. In any case and as long as the cipher used is secure, Eve, the adversary listening in on the communication, is unable to read the message since she does not have the key.

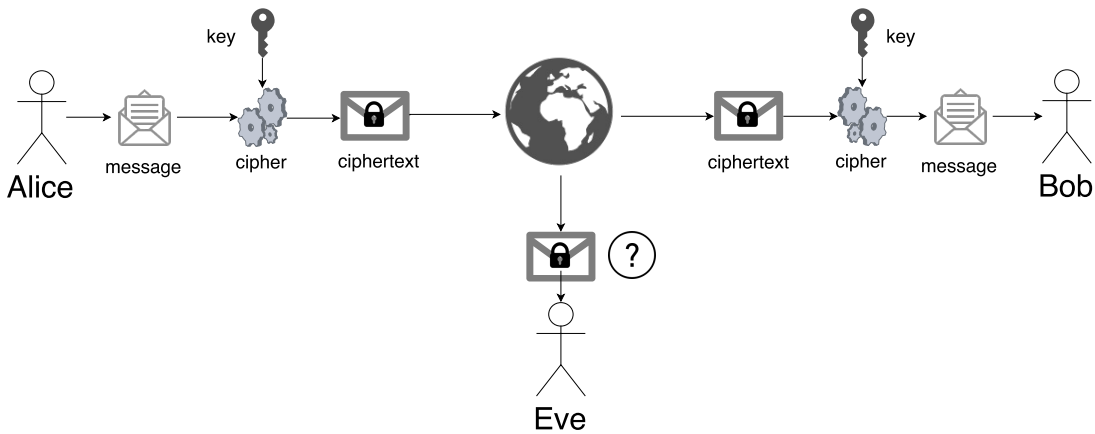

Figure 2.1: Communication flow in a confidential security service.

In order to develop these cryptosystems, there are two fields that dedicate themselves to studying them. The first studies how to achieve some security requirement, for example, in case of confidentiality, what scheme could ensure a message is confidential to third-parties. The field in question is

denominated cryptography.

Definition 3. Cryptography *is the field that studies mathematical techniques for conceiving cryptosystems.*

It is also important to study existing ciphers and know their security limitations. Techniques can be developed that exploit weaknesses in the cryptosystems, called *attacks*. This is the concern of cryptanalysis.

Definition 4. Cryptanalysis *is the field that encompasses the analysis of cryptosystems, in particular to defeat their implemented security requirements.*

Naturally these fields are complementary, for example, the cryptanalytic knowledge of possible attacks helps improve the design of ciphers in cryptography. These two fields can be said to belong to a more general area, cryptology (see Figure 2.2).

Definition 5. Cryptology *is the science concerning cryptosystems.*

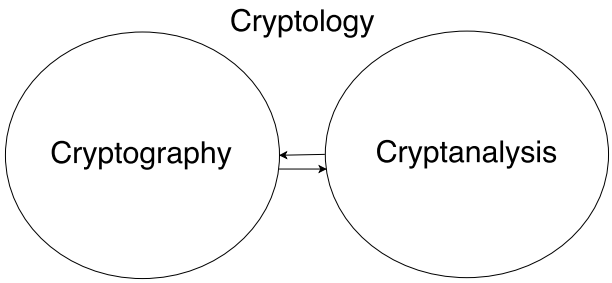

Figure 2.2: Domains of cryptological knowledge (based on Swenson [4]).

In classical cryptanalysis, a cipher can be considered as an abstract mathematical entity: the algorithm, its input and output. Mathematical design flaws can be encountered that reduce the security and help reveal the key and the message. If one is considering just the algorithm, the attack model is called a black-box model (see Figure 2.3). Since the black box model does not consider the cryptographic device at all, it is implicitly assuming it is not a source of security issues, or in other words, that it is implemented in some ideal secure device.

The notion of "black box" is commonly used to refer to when a system's implementation is not accessible. However, here "black box" is employed in the sense of a hidden *physical* implementation, but not a hidden algorithm — the algorithm and its constants are always assumed to be public.

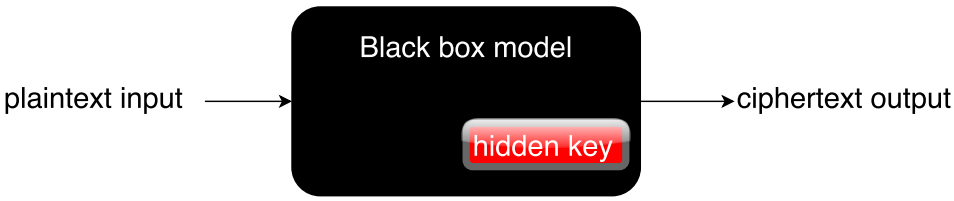

Figure 2.3: Diagram of a black-box model.

To find security weaknesses, it is important to clarify what we mean by secure and weak. Claude Shannon [2] first considered this problem and defined the concept of perfect secrecy:

Theorem 1. *Let M be a message and E the result of encryption (the ciphertext). Then, a necessary and sufficient condition for perfect secrecy is that*

$$P(E|M) = P(E), \quad \forall M, E \tag{2.1}$$

where $P(E)$ is the probability of the algorithm returning the ciphertext and $P(E|M)$ is the conditional probability that the algorithm returns the ciphertext E if message M is chosen. That is, $P(E|M)$ must be independent of M.

If the probability of choosing some ciphertext as the output of the algorithm does not depend on the message, there must be *at least* as many keys as messages, so as to transform every possible message into a different ciphertext. Such a cipher is called the *one-time pad*, where a message is added to a random key of the same size. To be clear, the key has to always be random, i.e., it cannot be reused (thus "one-time").

Notice the input message can be very large, and generating random keys of the same size is not feasible because sources of randomness are often not available or data is produced faster than the bits these sources can return. Therefore, keys are typically fixed to a given length (usually denoted by their bit length n), which means that there exist necessarily less keys (2^n) than input messages (∞).

Cryptanalysts do not usually analyze if a system is secure based on the concept of perfect secrecy, but rather on a relaxed definition with assumptions about the complexity of the cipher and its computability.

The first definition was introduced by Goldwasser and Micali [5] as *semantic security*. The name *semantic* refers to the fact that the encryption output should reveal no information no matter the meaning or structure of the original message [6]. In simple terms, encryption is considered to be secure if a third-party has a *negligible* probability of recovering information on the message from the ciphertext in a *reasonable time* [3, 7].

An attack is considered to be carried out in a *reasonable* time when it takes substantially less *computational complexity* than an exhaustive search of the key space. Searching for a key, i.e. guessing, is an operation that requires testing all the 2^n keys (binary keys of length n). If attacks of reasonable time are unlikely, this definition ensures that any weaknesses in the encryption scheme are asymptotically just as difficult as guessing. Similarly, *negligible* probability implies the probability of a reasonable attack must decrease asymptotically with some power of the key length.

Notice that this definition is similar but weaker than perfect secrecy, since it only gives an asymptotic bound instead of an actual guarantee of secrecy. It's still possible to make the key as large as necessary to approach the bound and ensure practicality and security, as long as all attacks take the same order of time as searching for the key. But how can one prove that?

In practice, it's impossible to evaluate exactly how difficult the algorithms are and assumptions have to be made. So, on top of semantic security, a new definition of *provable security* [8] was introduced, which not only considered the asymptotic approximation but also introduced assumptions about the hardness of the computability of some mathematical problems. Modern cryptographic schemes have bounds of

the order of the age of the universe for breaking a ciphertext if a classical computer is assumed, but quantum computing may be able to break them much faster.

However, one must not forget that even with a secure cipher the physical realization of such algorithm is a cryptographic device, typically implemented with electronics and thus not an isolated system, so the black-box model no longer applies. It needs power to operate, must accept some input commands and return the ciphertext using some digital interface, and inevitably will have losses such as electromagnetic emanations or can be subjected to faulty behavior. Due to these physical processes, information about the algorithm execution may leak, for example, in the power signal or in the electromagnetic fields radiated by the circuit. This is referred to as *leakage*. Cryptanalytic techniques applied by an attacker to these signals can exploit them and compromise the security of the device, often several orders of magnitude faster than searching for the key.

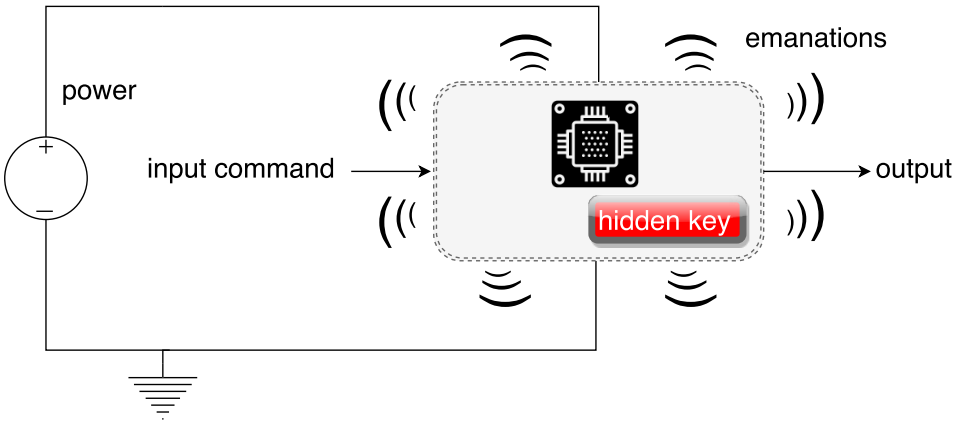

Figure 2.4: Model of a cryptographic device.

Attacks that compromise the *physical security* of a cryptographic device by exploiting their leakages are denominated *side-channel*, due to how they exploit information from channels other than the input and output of the device.

Definition 6. *A* side-channel attack *is an attack that exploits the data-dependent behavior of a cryptographic device to recover the device's key.*

The definition makes it clear that signals used in a side-channel attack have to leak information: if there is no data-dependency, then there is no information leakage. Leakage is in fact the result of physical behavior that depends on the data being manipulated by the device. This is the subject of the next section.

2.2 Leakage of cryptographic devices

Recall that cryptographic devices can either be implemented in dedicated hardware, perhaps with an application-specific integrated circuit (ASIC), or in general-purpose hardware, running on a processor. Either way, they all have an underlying electronic implementation with binary logic gates. The vast

majority of the electronic systems developed in the 21st century use complementary metal-oxide semiconductor transistor cells (CMOS), desirable for producing devices with low idle power consumption. In this section an overview of CMOS and its power and electromagnetic leakage will be given in order to give us the knowledge necessary to mount the attacks.

CMOS cells implement a logic gate with low power requisites by having a complementary pull-up and pull-down network that alternate in insulating the path between the power source and the ground, only briefly consuming power during transitions when momentarily both networks conduct and power is consumed.

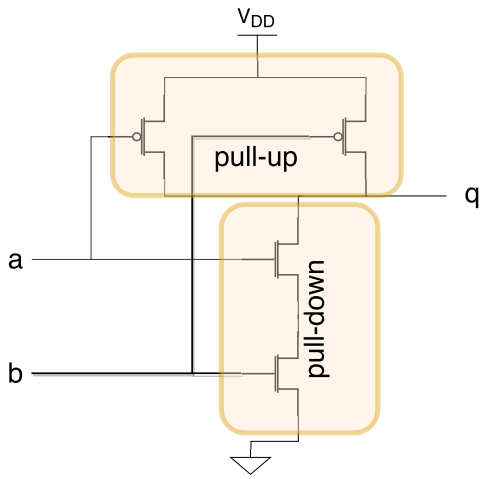

Figure 2.5: CMOS transistor implementation of a NAND cell [9].

Power consumption can then be characterized by two components: the static power consumption and the dynamic power consumption. In CMOS circuits, the former is low due to the aforementioned behavior, thus the latter is the main source of power consumption.

These electronic circuits are direct-current (DC) powered. To provide power to the cryptographic device, the generator works to keep the voltage constant, but the cryptographic device has time-varying power consumption which translates into a current variation. In order to measure the current, a small resistor ($\approx 1\ \Omega$ [9]) can be placed in series with the cryptographic device which enables monitoring of the current by measuring the voltage drop at the terminals of the resistor. That voltage drop is then an indirect measure of the power consumption. In fact, when referring to power consumption in side-channel attacks, one usually means some indirect measure of that consumption, such as the voltage drop in a 1 ohm resistor. The actual value of power consumed is often irrelevant, since the relative variations of measured power encode the information for the attack.

Electromagnetic emanations

After the previous discussion of the relationship between power consumption and current, an analysis of the relationship between current and electromagnetic emanations is warranted. By doing so, it follows that the electromagnetic emanations will also leak information that can be exploited with similar techniques as those used for power consumption.

To introduce this relationship, first consider the case of a stationary current (that is, one that does not vary with time). There is a well established theory on the relationship between static currents and the magnetic field they induce, a phenomenon first observed by Hans Ørsted. In fact, it can be said (Figure 2.6) that each oriented infinitesimal piece of wire (denoted as $\overrightarrow{d\ell}$), as it carries some current (denoted I) produces a contribution to the magnetic field (denoted dB, since it is a differential, i.e., an infinitesimal change in the field) at some point p. Notice that even though the current is flowing in some direction, current is typically denoted as a scalar, and the infinitesimal piece of wire is oriented instead. As such, a positive current is a current flowing in the orientation of the wire, and a negative current is a current flowing in the opposite direction.

This contribution depends on the distance and orientation to point p, which can be represented as a vector $\overrightarrow{r}$ from the piece of wire to that point. This vector has a norm, which is the distance to the point (denoted r) and the direction from the wire to that point (denoted $\hat{r}$, as in, the unit vector pointing in the direction of the vector $\overrightarrow{r}$). The law that established this relationship is the Biot-Savard law [10]:

$$d\mathbf{B} = \frac{\mu I \overrightarrow{d\ell} \times \hat{\mathbf{r}}}{4\pi r^2} \tag{2.2}$$

A new constant is also mentioned, the μ, which is a quantity that takes into account the medium where the magnetic field is produced, called the magnetic permeability. The cross product between the orientation of the wire (and thus, the current) and the position vector means that the direction of the field will be orthogonal to the two vectors.

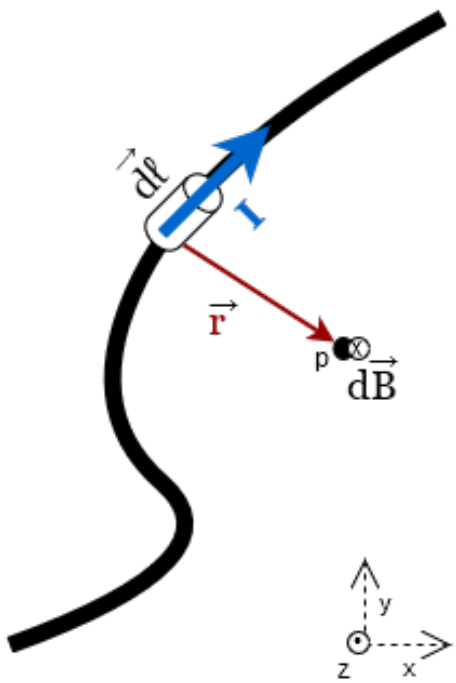

Figure 2.6: Illustration of the Biot-Savard law. The dot and cross convention in the referential indicate a vector pointing to outside and to inside of this page, respectively. The $\overrightarrow{dB}$ vector is meant to be at the same location as point p.

The integration of the expression along the path of the wire would sum all the contributions of mag-

netic field to that point. It can be seen that the magnetic field depends linearly with current, decreasing or increasing the field measured accordingly.

However, the field produced by a circuit is not a static magnetic field, due to current variations in time generated by the electronic operations (and subsequent variations of magnetic field). The approximation of the Biot-Savard law is still valid if the field is measured close to the source, called the *near-field*. In the near-field, the magnetic field decreases with the square of the distance, $1/r^2$, unless the source is so close (called the *reactive near-field*) that the magnetic and electrical fields are too complicated to model.

The reactive near-field decays rapidly (with powers greater than 3), so its influence is confined to a very small region, although it is important to be aware of its existence when measuring magnetic fields with a probe very close to the circuit. At these small distances the field can be so strong as to cause the probe itself to radiate back as it picks up the magnetic field, and thus interfere with the measurement. Also, since all elements of the circuit are within small distances of each other, the same phenomena applies: fields generated in the circuit can couple with fields emitted by other elements, which results in the modulation of these fields [11]. *Coupling*, as the name implies, is when the fields "pair up" and vary together. *Modulation* is the type of variation introduced: it refers to how the resulting signals are simply the original signal but affected by changes in amplitude or the angle of the wave due to the other signal's influence.

As a result, it may be possible to observe significant leakage by exploiting otherwise uninteresting signals, such as the clock signal, which may be modulated in either amplitude or angle with the coupling of other signals in the circuit.

Other region is of importance, the *far-field*. In accordance to Maxwell's equations, the varying current generates electromagnetic (EM) radiation which propagates out to the surrounding space. In the far-field, the electric and magnetic fields propagate as a wave, inducing each other, and thus varying not with the square, but with the inverse of the distance $(1/r)$. But at which distance can we consider the field to behave as a far-field?

An important property of the field is how frequently it varies: its frequency. The emanations that do not arise from coupling, i.e. that are generated directly by the elements, are called *direct emanations* and their frequency and power is related to the actual switching activity of the transistors. This means the power is typically very low and the frequencies tend to be much smaller than the clock frequency [11]. The *unintentional modulated emanations*, however, typically arise due to strong clock signals. A square clock signal can generate many high frequency harmonics, or multiples of the original frequency of the clock, which arise due to the quick changes caused by the sharp edges of the square wave. Then, unintentional modulated signals are typically high frequency, of the order of the clock signal [11].

The electromagnetic field propagates at approximately $c = 3 \times 10^8$ meters per second in air. This means the electromagnetic wave, as it varies in time with frequency f, will also vary in space as it propagates away from its source. If the signal is periodic in time, then the variation is also periodic in space as it travels away. The variation periodicity in space is represented by a distance called *wavelength* λ, and can be calculated by taking into account both the frequency and the speed of the wave in the

medium:

$$\lambda = \frac{c}{f}. \tag{2.3}$$

In fact, the near and far-field are defined by the wavelength, with the boundary typically set at $\frac{\lambda}{2\pi}$. For small frequencies, the wavelength is long, which means that direct emanations, with lower frequencies, lower power and typically measured close to the circuit, are near-field phenomena. For the modulated emanations, the frequency is large enough ($> 10^8$ Hz) to reduce the wavelength to the scale of meters ($\lambda < 3$ m). Due to their higher power, both near-field and far-field measurements are possible.

The field amplitude encodes the power consumption, and since the field is radiated from the chip, the attack can be set up to focus on the area of the chip that radiates the most leakage. Instead of capturing the total power consumption, focusing the EM field capture on certain areas avoids the influence of power consumed in parts which do not leak useful information and are 'noise' to our intended attack. For EM near-field capture, even a small coil of wire could suffice [10], for far-field capture, a larger antenna with considerable gain is required.

The orientation of the field depends on the current direction inside the circuit, so EM typically encodes more information [10] than the power capture thanks to changes of current flow inside the chip. Because these are local current direction changes across sections of the circuit, they do not influence the direction of the current at the power supply, and are thus information that can only be obtained in electromagnetic attacks.

In essence, EM attacks have the advantage of allowing the attacker to focus closer to the chip area where the field depends the most on the data, and encode not only the total power consumption but also the flows of current inside the circuit, which also leak information of the state transitions in the chip. On the other hand, as will be later discussed, the quality and power of the EM signal can make it significantly harder to attack this side-channel.

Operation and data-dependent consumption

As established, power consumption of CMOS circuits depends on the transitions of the logic gates which means it will depend both on the operations and the data manipulated by the circuit. These dependencies are crucial to carrying out the attack.

Regarding data-dependency, whenever data is loaded into a cryptographic device's bus, the transitions that it triggers in the CMOS cells are related to the change in bits between the original value and the new value. The change in bits between two values is referred to as the Hamming distance between two values, $HD(v_0, v_1)$, where v_0 is the initial value and v_1 is the new value. Because this is a binary system, it is mathematically identical to computing the XOR of both values, or a modulo-2 difference.

However, this presents a problem. An attacker prefers the power consumption to depend on the data, not on the difference to a potentially unknown value. In practice, this value may be known, for example, if the bus lines are reset to either '0' or '1' before loading data (called *precharging* in the latter case). If the bits are all '0', transitions in the bus will only occur for the '1' bits in the new value, whereas if all the bits are precharged to '1', then transitions will only occur for bits that are '0'. In the first case, transitions

happen for the number of bits that are '1' — that quantity is called the Hamming weight of a binary value. In the second case, exactly the opposite: the transitions occur for all the bits that are not '1', that is, higher Hamming weights in the new value will correspond to less transitions. In either case, the power consumption will depend only on the new data and not on the previous value, with a higher number of transitions corresponding to a higher power consumption.

To be clear, it does not matter if the power increases or decreases with Hamming weight, as long as the Hamming weight of the data is related to the power consumption. In practice, even if the previous value in the bus is not known nor fixed, there is always some information that can be obtained from a Hamming weight model for power consumption, with enough statistical processing.

If the previous value is completely random the data-dependency still exists [12]. This is due to the fact that the symmetric transition model previously employed is a simplification. Unlike assumed above, real power consumption is different for the $1 \rightarrow 0$ and $0 \rightarrow 1$ transitions [9, 12]. Since the power doesn't depend only strictly on the number of transitions but also on their end value (transition to zero vs. transition to one), there exists data-dependency on the final value. Of course, these transitions differences are subtle, but in EM attacks, which also capture the current flow between the transistors as they transition, the asymmetry is more evident.

So, for an attacker assuming the Hamming weight power model, if the bus is in the first case, precharged or set to zero, the attack is easier to carry out than if on the second case, an unknown value on the bus, which in turn is easier than the last case, a random value loaded before a new value on the bus.

In the particular case of the electromagnetic traces, the flow of the current changes the polarity of the observed signal. Thus, a Switching Distance model was proposed by Peeters et al. [12].

The Hamming distance model calculates the number of flipped bits between two values, with every flip increasing the Hamming distance by one. The Switching distance, on the other hand, takes into account the direction of the bit flips. A transition of $0 \rightarrow 1$ is considered a positive transition, and increments the Switching distance by one. On the other hand, a transition of $1 \rightarrow 0$ is considered a negative transition and it decrements the Switching distance by one.

Power consumption is also operation-dependent, since different operations will trigger transitions in the circuit that generate different power signatures. These signatures may be identified by an attacker in what is called *profiling*, enabling detection of the executed operations. If the implementation of the algorithm includes operations that depend on the key, then an attacker can use their signatures to identify the operations executed and deduce the correct key that triggered them. In fact, one can say that operation-dependent behavior is only useful if it is also indirectly data-dependent, thus fitting the definition of a side-channel attack.

Power and noise model

Besides the varying degree of accuracy of these models, their simplified form cannot account for all the physical phenomena that occurs inside the circuit. This can be accounted for as *intrinsic noise*, only one of the many different sources of noise that will influence the actual leakage measurements, such as

the measurement noise and external noise. In general, taking into account some model h, the power consumption P can be described as a sum of the contributions of all the gates g in the circuit processing values at instant t, affected by some normally-distributed additive noise $\mathcal{N}$ [13]:

$$P(t) = \sum_{g} h(g,t) + \mathcal{N}(t). \tag{2.4}$$

2.3 Modern ciphers

In this section, modern symmetric ciphers and their techniques will be discussed. Recall that symmetric ciphers need to scramble the input message into ciphertext in some definite way using their key. This process needs to be such that it is hard to revert by any user but the ones with a copy of the key. To achieve this, ciphers employ diverse techniques to scramble the message. There are two main types of symmetric ciphers: *stream ciphers* and *block ciphers*.

Stream ciphers (see Figure 2.7) operate one bit at a time: using an algorithm based on the key and the message itself, they employ a pseudorandom generator to create a *keystream* which grows to be as long as the message. The keystream is added (modulo-2, i.e., XORed) to the message, bit by bit, to produce the ciphertext. It's important to stress that the keystream is not the key: the pseudorandom keystream must be recalculated in the same way from the key at the destination so that the message can be recovered. The strategy used is similar to the one-time pad, but of course, due to the pseudorandomness of the keystream, this cipher is only a practical approximation, and is not perfectly secret as discussed in the concepts (Section 2.1).

Stream cipher

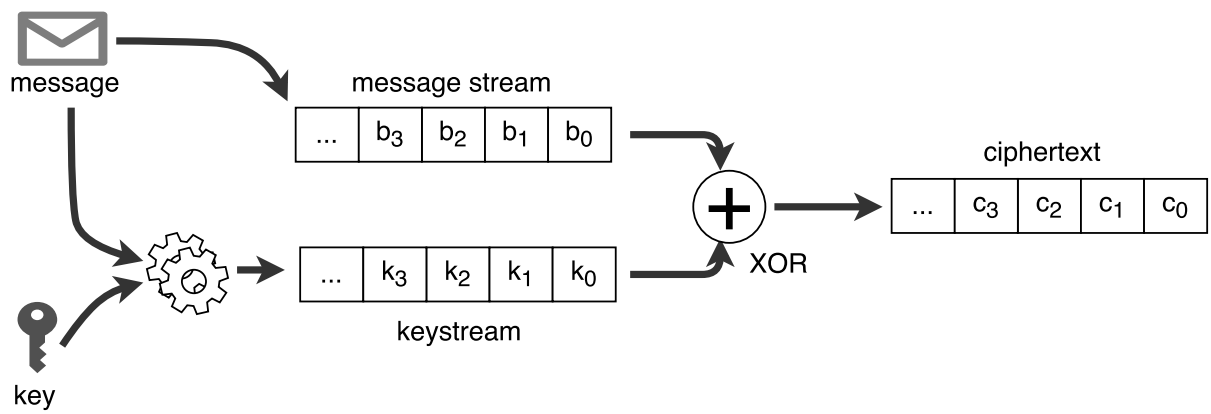

Figure 2.7: Diagram of a stream cipher.

Block ciphers (see Figure 2.8) operate on a block of the message bits at a time, processing all the bits in the block simultaneously and returning an output block. Each block of the message is scrambled independently by the cipher using substitutions and permutations of bits. Block ciphers are typically a cascade of several chained processing steps, called *rounds*, that repeat these operations to ensure good security. The block undergoing processing is called the *state*. Each round can, for example, take

the state, do some substitutions, some permutations, add the key and output a new state for the next round. Typically, all rounds are similar and the output is the end of the last round.

Resulting blocks are then assembled to produce the ciphertext. The simplest mode of assembling the blocks, *Electronic Codebook* (ECB), is just a concatenation of the encrypted blocks. However, since the cipher will output identical blocks for the same input block, this can reveal a pattern in the ciphertext whenever there is repetition in the original message. As such, other encryption modes are used in practice. [1]

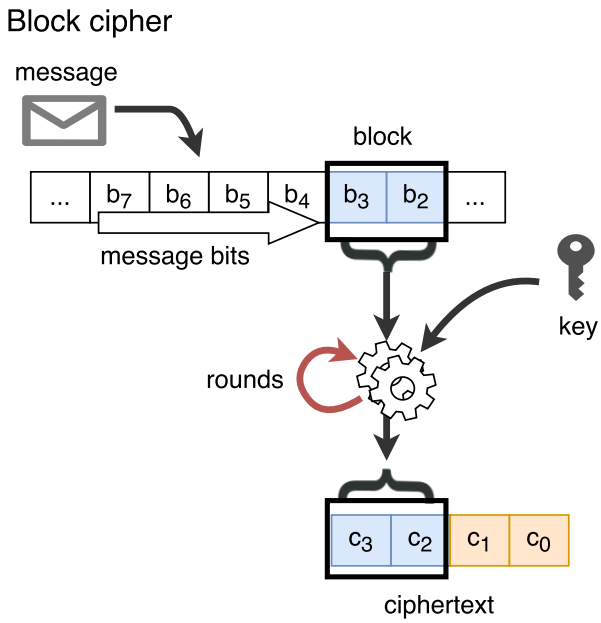

Figure 2.8: Diagram of an example block cipher (for 2 bit blocks, assuming the simplified ECB mode).

Block cipher operation can be divided into several logical blocks (see Figure 2.9). An *sbox* is the substitution element inside the round that looks up each input bit, checks its replacement in a standard lookup table and does the substitution.

These sboxes cannot do random replacements since it would be impossible to revert the process to retrieve the original message; however, the substitutions can be defined to be as arbitrary as needed as long as they are well-defined and standardized to allow decryption: the sbox will replace the values depending on the input, and should always replace the same way for the same input. In practice, the substitutions will be specified according to some function with certain cryptographic properties, in order to ensure the resulting cipher is secure. An sbox operates on a small number of input bits (typically several sboxes are used) and can be implemented as a lookup table.

A *permutation layer* is a layer that switches bit positions. Again, this scrambling has to be well-defined so as to be able recover the original message — switching the position of the bits randomly wouldn't work. This step can also be denominated as *wirecrossing*, since one can visualize each bit as an individual wire signal, and crossing the wires would switch the bit positions.

These two elements can be combined into a *substitution-permutation network*. In each round, the

[1] The modes of operation of block ciphers will not be detailed in this dissertation. The reader is referred to Menezes' Handbook of Applied Cryptography [3] for further reading on this subject.

network increases *confusion* (every bit is hard to trace) and *diffusion* (every bit is affected by many).

Instead of always adding the same key on every round, the *key scheduling* process derives different keys for each of the rounds from the original key, called *round keys*, that are then added in each round into the state.

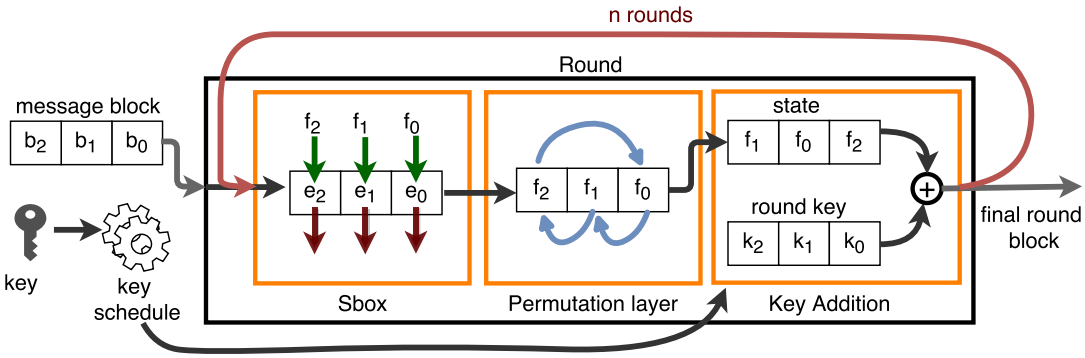

Figure 2.9: Block diagram of a simplified block cipher.

Advanced Encryption Standard (AES)

The Advanced Encryption Standard, originally defined as an American Federal Information Processing Standard (FIPS), is the *de facto* global standard [14] in encryption, used in all major applications due to its performance and security. It is a block cipher that operates on blocks of 128 bits (16 bytes) during 10 rounds. Since most operations are done on a byte-by-byte basis, a common representation of the state is a 4×4 matrix each element containing one of the 16 bytes. Four operations are executed in the rounds:

1. SubBytes

   This operation corresponds to sboxes applied to every individual byte. Each of the 16 bytes is replaced by another according to the sbox specification. A table is given in the standard which indicates the substitutions depending on the original byte.

2. ShiftRows

   This operation is a permutation that involves shifting the rows of the matrix using left rotations (i.e., removing the elements at the left and placing them at the end of the row). The first row is not shifted, the second row is shifted one element, the third row is shifted two elements, and the fourth row is shifted three elements.

3. MixColumns

   Each column of the state is multiplied by a matrix in order to combine the elements and obtain another column. This operation is done on Rijdael's Galois field using the matrix in the algorithm specification. It works in tandem with ShiftRows: their combined result permutes both rows and columns to ensure confusion, and in particular, MixColumns computes its result from the values in each column, creating diffusion.

4. AddRoundKey

   This operation corresponds to the round key addition to the existing state, as previously discussed in the block cipher overview. In XOR additions, every bit position of the output is affected only by the corresponding bit positions in the input. This means that each byte of the state will only be affected by a corresponding byte of the round key - that byte is commonly referred to as a *subkey*.

The round operations are executed in the order given, but there is an extra AddRoundKey operation in the beginning of the cipher with the original key (instead of a round key), and the final round does not include the MixColumns operation. So, in conclusion, a pseudo-code is presented with an overview of the algorithm execution.

Algorithm 1 Pseudocode for the Advanced Encryption Standard.

```
function AESCIPHER(byte input[4][4], byte key[4][4])
    byte state[4][4]                                        ▷ Declarations
    byte roundkey[10][4][4]                                 ▷ (10 round keys 4x4 bytes)
    byte output[4][4]

    state := input
    state := ADDROUNDKEY(state, key)                        ▷ Initial key addition
    roundkey := KEYEXPAND(key)                              ▷ Compute round keys

    for round := 1; round ≤ 10; round++ do
        state := SUBBYTES(state)
        state := SHIFTROWS(state)
        if round != 10 then                                 ▷ MixColumns is skipped in the last round
            state := MIXCOLUMNS(state)
        end if
        state := ADDROUNDKEY(state, roundkey[round])
    end for
    output := state
    return output
end function
```

The following section will elaborate on side channel attacks in view of the knowledge of modern ciphers and the leakage produced by the cryptographical devices.

Chapter 3

State of art

Recall the “side-channel” in side-channel attacks is any leakage source that can be exploited to an attacker’s advantage other than the usual plaintext input and ciphertext output of classical cryptanalysis. Many side-channels have been found over the years.

The first side-channel attacks were *power analysis*, introduced by an article in 1999 by Kocher et al. [1] with the observation that there was a statistical dependency between the power signal and the values calculated by the cryptographic device, which allowed the attackers to infer information about the key.

The same article ([1], Section 7, “Related Attacks”) stressed that other physical processes can be exploited in the same way, notably electromagnetic emanations. Whichever the medium used, emissions from a device, whether power fluctuations or EM emanations, all leak similar information about the physical computation performed by the circuit and so their cryptanalysis is similar, even in less common channels, such as in the acoustic attacks by Genkin et al. [15]. As such, electromagnetic and acoustic attacks can be considered a *particular* case of power analysis. There are other side-channel attacks that are not related to emanations, but to timing differences or faulty behavior, and as such require different techniques which do not concern the goals of this work.

3.1 Classification of side-channel attacks

Generally, it is possible to classify side-channel attacks by the channel they exploit: *power analysis*, which concern power consumption and *electromagnetic attacks*, which use the electromagnetic signal emitted by the device.

Side channel attacks are also distinguished by how they are carried out: regarding the attacker activity, one can distinguish between two types of attacks.

Passive attacks are those where the device is operated according to the intended usage. This type of attack will monitor the leakage whilst operating the device in conformance to the specification. Power analysis is an example of a passive attack.

On the other hand, *active attacks* imply the device is manipulated in order to facilitate the extraction of

the secret key. The device could be made to operate outside intended usage by, for example, providing faulty input. Fault attacks are active attacks.

Regarding the attacker invasiveness, attacks can be considered *invasive*, *semi-invasive* or *noninvasive*. An attack is typically considered invasive if it resorts to electrical contact with the chip surface. Expensive equipment can be used to finely monitor data signals flowing in the chip or to create faults on them. Semi-invasive attacks [9, 16] are those where the device is *depackaged* but the chip remains intact, i.e. no electrical contact with the chip surface is made. Finally, noninvasive attacks ensure the device is left physically intact: no physical evidence of the attack can be obtained beyond regular handling of the device, and the equipment used is usually inexpensive.

As previously mentioned, power analysis attacks are carried out passively and are typically noninvasive (semi-invasive attacks are possible if monitoring the power of a specific part of the circuit with a probe). They require several recordings of the power leakage from the target device over time, these are designated *traces* and can be obtained using standard signal acquisition equipment. A trace is captured over the period when the calculations of interest are being carried out by the device. For example, to attack the first round of AES, one can capture the power that leaks as the round is executed, and do so several times in different conditions in order to gather enough information. Recall that these attacks can not only exploit a power signal, but also EM emissions, so the same concept applies for EM traces.

Power attacks are concealable due to their low invasiveness, pervasive due to the abundant sources of leakage and relatively cheap to carry out using standard equipment. This grants them special interest in the cryptanalytic community. The process of analyzing power signals can be classified depending on the techniques used to extract information, typically in two different ways: Simple Power Analysis (SPA) and Differential Power Analysis (DPA).

3.1.1 Simple Power Analysis (SPA)

SPA consists in the usage of simple techniques, such as visual inspection, in order to retrieve the key. Some devices have leakages that include certain patterns clearly visible in a trace, these may enable an attacker to find the correct key by mere visual inspection.

SPA is simple due to the fact it forsakes processing and uses a small number of traces, perhaps only one, typically analyzed over the time domain. Kocher et al. [1] define it to be "*a technique that involves directly interpreting power consumption measurements collected during cryptographic operations*". Typically, it uses techniques that may require specific knowledge of the operation of the device, but no complex statistical processing nor specific algorithmic calculations. Operation-dependent traces are well suited for SPA since operation signatures are typically visible on a time graph, and the sequence of operations over time may leak the key.

Even if not employed as a proper attack, SPA techniques can be used to gather important information about a cryptographic device to later employ a more sophisticated attack, such as DPA. For example, identifying the operations executed allows the attacker to locate the operations being executed on the device and to target the attack to one of the operations.

3.1.2 Differential Power Analysis (DPA)

DPA is differential due to how it compares differences between measured traces and a hypothesis computed with a model of power consumption. It uses statistical analysis so it can reveal a key under considerable noise and without specific knowledge about the cryptographic device, analyzing over a large number of traces but on a small set of time samples, perhaps one. For this it requires a model either informed by the theory of power consumption in cryptographic devices — such as the Hamming weight model previously discussed — , or built statistically with the cryptographic device in question — a *template*.

In simple terms, a differential attack starts by capturing and storing the power consumption of a physical device encrypting a given set of known plaintexts. The remaining steps can be carried away far from the device.

The attack proceeds by taking a working hypothesis over a small part of the key — for example, "is the value of this partial key the number 129?" — and, using the same plaintexts as the physical device, it computes the intermediate calculations of the cipher state. From the intermediate values, using a model of the power consumption of that device, the expected power consumption for those cipher operations is computed.

Then, the differential attack checks if the expected power consumption agrees with the observed signals from the physical device — for example, when the values calculated cause the expected power consumption to increase, does the actual observed consumption also increase? If the differences in the expected power follow the actual power consumption, one has good evidence that the working hypothesis is correct.

Using a statistical test, it is possible to quantify this agreement so that the partial key chosen is the one that displays the best agreement of all tested. This is called the *distinguisher*, because it is the metric that allows the attack to *distinguish* the correct secret from the remaining (rejected) hypotheses.

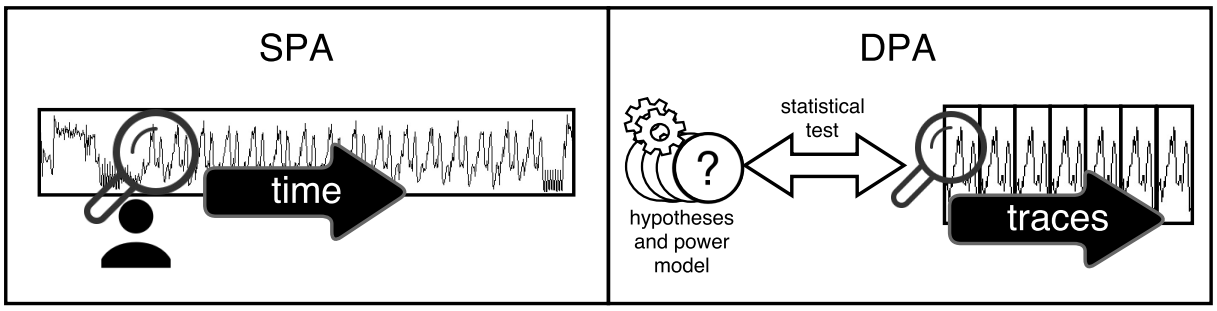

Figure 3.1: Comparison between SPA (for one trace) and DPA techniques. Example trace used in the diagram taken from [1].

One may ask: why not simply model the power consumption of the hypotheses for the key? Using an intermediate value may seem an unnecessary step, but is in fact essential for the attack to work. Recall that the key the attacker wants to recover is fixed secret stored inside the device, such as a smartcard. If the key is always the same, a model of the power consumption will always compute the same power hypotheses, since the values of the key are not affected by the plaintext input.

By using an intermediate value that depends not only on the constant key, but also varies with different input, the power measured in traces will vary as well: each captured trace will be slightly different at the instant of the manipulation of the intermediate value. These differences are important, since the hypotheses will have to be able to replicate them in order to agree with the measurements, which will only happen when the hypotheses are made with the correct key. This is the reason why a good intermediate value often involves a calculation with an sbox: a substitution step creates significant differences between runs that the attack can exploit.

Notice that making an hypothesis of the entire key value is not efficient — this is no better than a simple exhaustive key search, since all hypotheses have to be tested. Then, an intermediate value is chosen so that it depends only on a few bits of the key. All DPA attacks apply a divide-and-conquer strategy [10]: the attack is only performed on part of the key, and iterated over all the parts until the entire secret is recovered. This reduces the time complexity of the key hypothesis phase from the exhaustive search 2^n to $k \cdot 2^{\frac{n}{k}}$, (with *n* the bit length of the key and *k* the number of partitions), turning a complex problem into k simpler problems.

Briefly, the process can generally be described by the following five steps [9], depicted in Figure 3.2:

1. Choosing an intermediate result of the algorithm and determining how to calculate it from known data, such as the message or the ciphertext;
2. Capturing the traces and corresponding known data;
3. Making hypotheses on the partial key, and calculating the corresponding intermediate values of the traces using the known data;
4. Modeling power consumption based on the hypothetical intermediate values;
5. Comparing the hypothetical power consumption with actual power consumption using a statistical test.

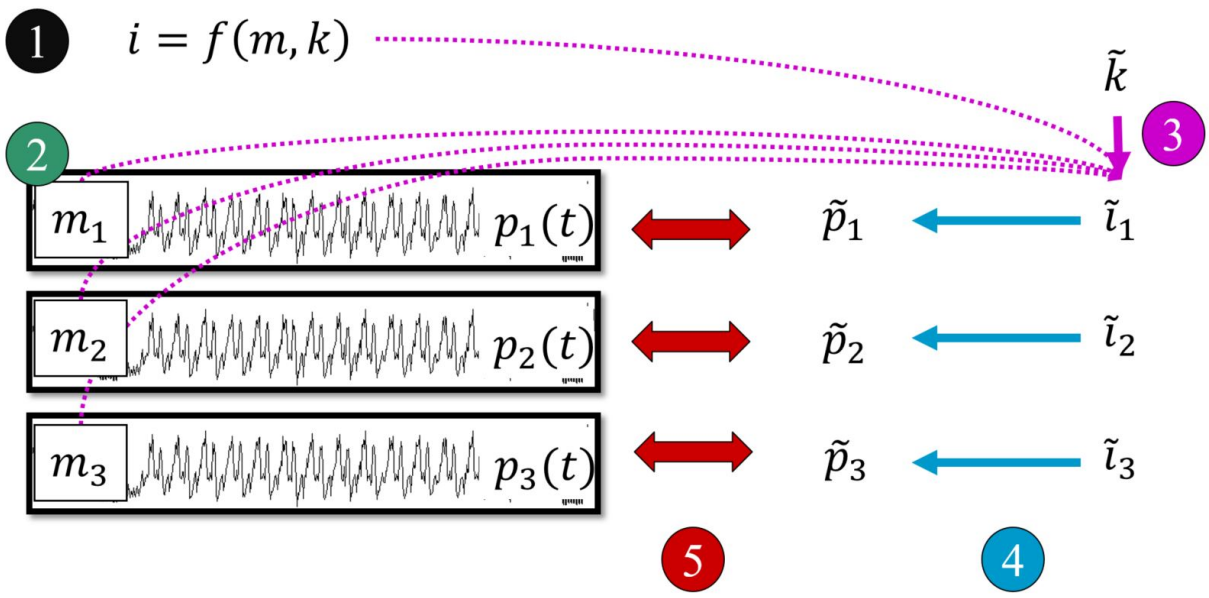

Figure 3.2: Illustration of the several steps in a differential attack. Example trace used in the diagram taken from [1].

All DPA attacks share these steps but vary on the number of bits attacked, the model used and the statistical test chosen to compare the model with the traces. As per Mangard et al.'s [9] nomenclature, DPA can be classified by their *distinguisher*, even if they are often introduced by their authors as entirely new attack classes. The main DPA techniques are:

— Difference-of-means attack;

— Correlation attacks;

— Template attacks.

In practice, it's not always possible to know beforehand which time sample(s) correspond to the manipulation of the intermediate value chosen, so the attacks produce *differential* traces, which show the value of the distinguisher over time for the key hypothesis in study. Then, the time instant that yields the best distinguisher value is chosen, which is then the instant where one expects the manipulation of the intermediate value to occur. Each key hypothesis is then compared according to their distinguisher, and the key is revealed. To continue, the specific DPA attacks will be elaborated further in the following sections.

3.2 Difference-of-means attack

First introduced with a groundbreaking paper by Kocher et al. [1], the original differential power attack (DPA) used the difference-of-means test for a six-bit attack on the DES cipher using one leakage bit of an intermediate value. This text will not elaborate on DES, but will provide a simplified framework to understand the original authors' reasoning.

Recall the first step of a DPA attack, choosing an intermediate value and finding how to calculate it from known data. To carry out the attack, Kocher et al. selected the beginning of the last round as their intermediate value - i.e. the calculation chosen happens just before the last operation using a key of the DES algorithm, which then returns the ciphertext. This means that the last round of DES involves some round function R that operates on the domain of an intermediate value L and the round key value K, returning the output ciphertext C, defined as $R : f(L, K) \rightarrow C$.

The goal is to be able to calculate the intermediate bits from the output ciphertext, which is known. This is called a known-ciphertext attack. Now, define a function D' that computes the intermediate value from the ciphertext and the round key, $D' : f(C, K) \rightarrow L$, akin to reversing the last round. However, instead of returning L, let us instead focus on each of the bits of the L value, since in DES different bits are processed by different sboxes and affected by a corresponding partial key. For DES, the intermediate value L is 32 bits long. The function to be used, denominated DPA selector function D is the function that returns the value of a bit of L given C and the partial key K_s, and is defined as $D : f(C, b, K_s) \rightarrow D_i$ where b is the bit selector in the interval $0 < b \leq 32$.

The attacker begins by collecting m traces of $k - 1$ samples $T_i[0...k]$, $i \in [1, m]$. The cryptographic device is given some random input, to which it returns the corresponding ciphertext C_i while the capture device records T_i. The pair (T_i, C_i) is stored for each of the runs. The attacker then chooses a bit to target: different bits of the intermediate value are affected by different partial keys. No explicit power hypotheses will be made, only the simple assumption that 1 and 0 in the leakage bit represent a difference in power consumption values that can be measured and used to infer the associated partial key. Now, the attacker knows C_i, b but wants to know K_s.

Since this is a partial key, the exhaustive search is much faster due to its smaller size. The attacker makes hypotheses on K_s, and then takes every (T_i, C_i) pair and computes $D_i = D(C_i, b, K_s)$. If the $D_i = 1$, then the trace is placed on the set of m_1 traces with leakage bit 1, $S_1[1...m_1][0...k]$, otherwise, it is placed on the set of m_0 traces with leakage bit 0, $S_0[1...m_0][0...k]$. Notice that if the partial key guess is wrong, the assignment of traces will be mostly random.

A mean trace of each set is computed, the $M_0[0...k]$ and $M_1[0...k]$, and a differential trace is calculated from the difference of the two, $\Delta_D = M_0 - M_1$. If the traces were grouped randomly, then the difference of the averages will approach zero ($\lim_{m\rightarrow\infty} \Delta_D \approx 0$), but if the partial key was correct, the averages will differ ever so slightly due to the bias of the leakage bit, which will show as a spike in the differential trace. It's important to mention that although the traces were captured independently, they must be aligned properly in time domain in order for the mean traces (and the differential trace) to make sense. All the steps involved in this attack can be computed by a single function on the traces, using the DPA selector function to calculate the average of the corresponding traces and then subtracting them, as expressed in the original paper:

$$\Delta_D[t] = \frac{\sum_{i=0}^{m} D(C_i, b, K_s) \cdot T_i[t]}{\sum_{i=0}^{m} D(C_i, b, K_s)} - \frac{\sum_{i=0}^{m} (1 - D(C_i, b, K_s)) \cdot T_i[t]}{\sum_{i=0}^{m} (1 - D(C_i, b, K_s))}. \tag{3.1}$$

Finally, the attacker will run this several times for different leakage bits b_n in order to reveal the

different parts of the round key and complete it. In the end, the attacker should be able to compute the original key from the round key. Notice that if the hypotheses are made only on one bit of the intermediate value, which is manipulated only at some instant in the traces, not all of the time instants are relevant. If *a priori* the attacker is reasonably confident on the time instant where the bias of the intermediate value is located, the traces can be trimmed to include only that instant. The difference of means can be calculated only on those time samples and the right key would correspond to the highest bias observed. Other variations of this attack were later developed, including the so-called 'distance of means' attack [9] which also took into account the sample variance.

3.3 Correlation attack

In the previous section, traces were classified in order to reveal the influence of one leakage bit, by subtracting the means of the two sets. However, in general, leakage is affected by several of the bits and it would be useful to use as many of them as possible, affected by the same key, so as to improve the attack in light of the more complete models already presented in the leakage discussion (Section 2.2), such as the Hamming weight model.

A first approach would be to partition the traces not according to one leakage bit, but the Hamming weight of the intermediate value. In any case, this approach, referred to as a *Multi-bit DPA* [17], is still limited in performance. These classes of attacks are sometimes referred to as *Partition Power Attacks*.

Instead of comparisons between partitions, ideally, the attacker may wish to have some *metric* of agreement between the hypothesized power model of the intermediate bits and the fluctuations of power consumption. This attack model was first introduced by Eric Brier et al. [18] and achieved just that. In correlation attacks, this metric is the estimator of the Pearson correlation coefficient, which will indicate a linear dependency between the hypotheses for each trace h_i and the power consumption observed at some given time instant t_0:

$$r_{i,t_0} = \frac{\sum_{i=1}^{M}(h_i - \bar{h}) \cdot (T_i[t_0] - \overline{T[t_0]})}{\sqrt{\sum_{i=1}^{M}(h_i - \bar{h})^2 \cdot (T_i[t_0] - \overline{T[t_0]})^2}} \tag{3.2}$$

where the means are indicated by a bar.

Carrying out the attack is in itself very similar to the general framework already introduced: starting with choosing the intermediate value, executing the trace capture, calculating the intermediate value based on a key hypothesis, calculating the power consumption estimate based on the intermediate value, and finally using the statistical discriminator to evaluate the quality of each hypothesis and find the right key.

An example 8-bit AES attack to the first round is thus given. The intermediate value chosen is the value after the first sbox. To calculate this intermediate value, recall that the AES algorithm starts by adding the original key and then performing the SubBytes operation, which corresponds to doing substitutions (sboxes) to every byte in the state. Focusing only on a single byte of the input plaintext P_i, the intermediate byte I_i is calculated by a simple function $I_i = SboxAES(P_i \oplus K_s)$, where K_s is

the corresponding key byte. Recall that $SboxAES(x)$ is part of SubBytes in the AES algorithm and is publicly specified.

As before, the device captures the trace $T_i[0...k]$ and some information to help calculate the intermediate value for that trace. In this example, for the first round of AES, the input of the cipher is needed — the plaintext P_i. As before, m pairs (T_i, P_i) are stored, $i \in [1...m]$.

An hypothesis on the key byte $\tilde{K_s}$ is made. Based on this key, the intermediate values $\tilde{I_i}$ corresponding to each trace can be computed, and calculating their Hamming weight gives a (relative) estimate of power consumption h_i. The values of the hypotheses do not need to accurately describe the power consumption because as long as they are linearly related to the power, the Pearson coefficient will capture their correlation with the traces.

Finally, in the time sample t_0 where the intermediate value is calculated in the circuit $t_0 \in [0...k]$, the estimator of the Pearson correlation is calculated between the set of traces at that instant $T_i[t_0]$ and the power model hypotheses h_i, $i \in [1...m]$. The process is repeated for each key hypothesis, and the key that yielded the best (highest) absolute correlation is the key byte that the attacker is looking for.

Often, the correct key power model will not correlate directly with the leakage traces — depending on the device architecture, in certain operations, the power consumption may instead correlate inversely with the power model. For this reason, and to maximize the information extracted from the attack, the absolute value of the correlation is used, which is highest whether the model is directly or inversely correlated.

Should this time instant not be known precisely, as it often isn't, the correlation can be calculated for several time samples $\tilde{t_n}$ across the traces and the best (highest) absolute correlation $|r|$ is recorded out of all the time samples considered. In a sense, a *correlation trace* $r_D[t]$ is created which indicates the correlation over time for a given key. Since our model is only concerned with a specific intermediate value, the correlation will only be relevant in specific time instants where that value is manipulated. As such, the highest absolute correlation out of all the correlation values will be observed in the correct correlation trace for the correct key, at the correct instant where the intermediate value considered is being processed.

3.4 Template attack

The attacker may be interested in compromising some device, but may have limited access to it, only managing to capture a few traces of its execution. If the attacker is able to obtain an identical replica of that device — perhaps the device is mass marketed and wildly available - how could the attacker use it to gain additional information to help their attack?

The attacker can leverage the similarity between the two devices — the one the attacker owns and the one they intend to attack — in order to build some model of how the device leaks information on its calculations. This model can then be used to compromise the intended device with a small amount of captured traces.

Template attacks are an important class of side-channel attacks which take advantage of a pre-

existing empirical model to reduce the number of traces needed from the device under attack. In simple terms, a model is built using a device under the control of the attacker, using as many samples as needed. Since the attacker controls this device, the model can be built up to a substantial accuracy.

For the empirical model to hold, it must be built based on a fairly identical device in similar conditions. When this is true, the model offers the advantage of encoding more information than the original theoretical model used, since it is also indirectly modeling such effects as its construction and the circuit layout, which the theoretical model disregards. If, however, the device used for the model is different of the device attacked, the additional effects modeled effectively work as noise, since they are irrelevant to the device in study, in which case a theoretical model would likely yield better results.

There is an apparent trade-off: a considerable number of traces are required to create better models, to the extent that increasing the number of traces used in a model will then require less traces to attack the target device. This is useful for a potential attacker, since it potentially reduces the *online* attack time, which is the time that the attacker needs to be next to the attacked device for capturing traces. A shorter online time means, then, a stealthier attack which would likely go unnoticed.

The separation of an attack into two phases was first proposed by Paul Fahn and Peter Pearson [19] in 1999:

1. *Profiling* or *offline* phase

   The phase of the attack where the power consumption of each operation is characterised. For example, in AES, this is the consumption of each of the 16 bytes of the SBOX output for each of their possible 256 values (2^8) in some given round. This is done by the attacker using their replica of the device, far away from the real device that one intends to attack.

2. *Attack*, *key extraction* or *online* phase

   The phase of the attack where the power consumption of the attacked device is compared to the models built in the previous phase. For well-built and powerful models, the number of traces needed from the device under attack can be very small for a successful attack to all the subkeys.

In this case, instead of hypothesizing the intermediate value's power consumption based on the theoretical models, the attacker first carries out cipher computations on the experimental device. As the attacker has complete control of this device, there is no uncertainty pertaining the intermediate values calculated by it. For each intermediate value considered, traces are generated to capture samples at the time instant where this value is manipulated.

The attacker can now use a statistical metric, such as the mean, to model the consumption of each possible value of the intermediate calculation. The attacker will then repeat this for all the remaining intermediate values. For example, the attacker may arrive at the conclusion that the manipulation of the intermediate value "6" for that calculation (and at that time instant) generates power consumption measurements centered in 20mV (corresponding to the voltage drop of the resistor). After proceeding with measurements for all values, the attacker can then replace the previous Hamming model to achieve better performance in the previously described attacks.

Multivariate normal templates

So far in this work, *template attacks* were discussed as replacement model for power consumption, improving on the approximations made by theoretical models such as Hamming weights. However, empirical modeling of the device's power consumption allows for much improved attacks if one considers more advanced statistical techniques. This is what usually is meant as a *template attack*, a concept first introduced by Suresh Chari et al. [20] in 2002 based on Signal Estimation theory.

In building a template of a certain intermediate value, the attacker is characterizing the power consumption of that intermediate value in the cipher, by fixing that intermediate value in the device, in order to obtain several samples of that intermediate value's power. Recall the leakage model previously introduced for power attacks in equation 2.4, comprised by a sum of a data-dependent term and noise term, the latter normally distributed. To build a template for a certain intermediate value, the circuit is always calculating that same intermediate value in order to estimate its power. Notice that, for some invariant (fixed) manipulated value, the data calculated is the same, and thus the data-dependent term in the leakage model is a constant. The result of the sum must then be normally distributed due to the noise term. For this reason, templates are usually modeled using a normal distribution.

The model is obtained by first selecting an interesting time instant, or *point of interest*, where the value of the calculation being attacked leaks the most. In that time instant, one takes the average value m as an estimator of the distribution mean μ and the sample variance s^2 as an estimator of the variance σ^2 for the power consumption. The process of building an univariate template will result in (m_i, s_i^2) pairs, one for each of the possible intermediate values. In a sense, the mean represents the estimation of the data-dependent part of the leakage – the constant term in the leakage model –, while the variance accounts for the noise [21].

In fact, it is possible to use more than a single time point to construct a template model. A multivariate distribution allows the model to capture several instances of manipulation of the same value, providing a considerable improvement in attack performance [20]. If considering a multivariate normal distribution, it is possible to capture the dependencies on several interesting time instants (*points of interest*) to the intermediate value in the same model. The template building is still mostly similar to the univariate case: the average value is now a vector M_i, and the sample variance is replaced by the variance-covariance matrix Σ_i (also called a covariance matrix). A multivariate template consists of pairs of (M_i, Σ_i) representing the modeling of each operation, coherent with the univariate description.

More recently, the number of templates has been reduced when combining the theoretical power models and the empirical data, as proposed by Moulay El Aabid et al. in 2007 [22]. For example, since it is known that a good approximation of the power consumption is the Hamming weights model, one could assume that all of the values that have the same Hamming weight will have similar templates. Then, instead of calculating 2^n templates for an $n-$bit value, as an approximation it suffices to calculate $n+1$ templates for each of the possible Hamming weights, $0, ..., n$. Given that the number of traces needed to acquire a representative sample of each of the possible 2^n values is very large, this technique can reduce the complexity of the template profiling phase considerably, at the cost of reducing the quality of the model.

Maximum Likelihood Bayesian distinguisher

Building a multivariate normal template enables the attacker to establish relationships between different time points in the trace for the same intermediate value. However, conventional distinguishers, such as in the correlation attack, consider each time sample to be independent and, for that reason, cannot exploit all the information in multivariate templates. A different distinguisher is needed in order to fit this model into the observed data. The Maximum Likelihood Bayesian consists in the calculation of the likelihood of the observed power leakage in the traces, which is done by applying the Bayes theorem, improving the performance of the attack. This distinguisher is based on Elisabeth Oswald and Stefan Mangard's work presented in 2006 [23].

Intuitively, one knows the that, for every template, the probability density function, or distribution, indicates how the power consumption is distributed for the intermediate value of that template. A peak in the distribution corresponds to a measured power consumption that is very likely for that intermediate value, and, accordingly, a small value in the distribution indicates a low likelihood for that measured value.

From this intuition, the probability density function can be taken as the likelihood when viewed as a fixed function that depends on a parameter, in this case, the power consumption leakage ℓ. This is represented mathematically as $p(\ell|i)$, as the distribution of the power consumption leakage ℓ for some intermediate value i. As such, every template corresponds to a pair (M_i, Σ_i), which encodes a normal distribution $p(\ell|i) = \mathcal{N}(\ell|M_i, \Sigma_i)$.

To explain the Bayesian interpretation of this problem, consider that, when initiating the attack, all the possible intermediate values have the same probability of being correct, described by an uniform probability density $p(i)$. All intermediate values are considered equiprobable merely due to the attacker's ignorance *a priori* — one of the values is in fact correct for each calculation of the attacked device, but the confusion and diffusion properties of properly designed ciphers hide this knowledge from the attacker. To find the correct intermediate values, evidence will need to be collected in the form of traces.

As part of any differential attack, hypotheses are made on the intermediate values based on some hypothesis of the key. Which of the hypotheses are better supported by the measured power consumption leakage ℓ_m? The likelihood of the measured leakage is, taking the template of the hypothetical intermediate value i, the density of the distribution for the measured power leakage — $p(\ell = \ell_m|i)$. If the likelihood is high, the template of the intermediate value shows agreement with the leakage, and the intermediate value hypothesis is likely right; otherwise, the measured leakage is not supporting the hypothesis of the intermediate value that was made.

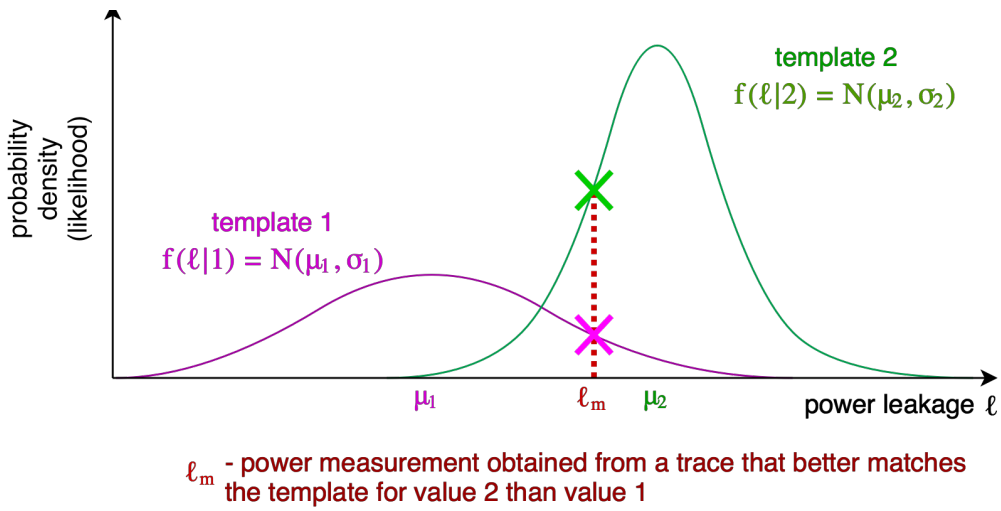

Figure 3.3: Illustration of the matching of the observed leakage to templates of two different intermediate values. For the power measurement in question, an hypothesis of a key that leads to the estimation of the intermediate value as 2 should have a higher likelihood of being correct than another key that leads to an estimation of this value as 1.

Taking the previous $p(i)$ probability for the intermediate value, where all the intermediate values were assumed equiprobable, one can update that probability by multiplying the newly calculated likelihood from the previous step, $p(\ell|i)p(i)$.

By weighting the likelihood $p(\ell|i)p(i)$ by the likelihood of the leakage, $p(\ell)$, the Bayes theorem allows one to derive a *new* probability density of the intermediate value, *but taking into account the observed leakage values, and how likely the leakage is*. This corresponds to the $p(i|\ell)$, which is the probability of the intermediate value i having observed leakage ℓ. It's important to stress these as *likelihoods*, and not probabilities, since they are points on a continuous distribution.

In summary, to model how some observed leakage ℓ could support the evidence of the intermediate value i, consider the Bayesian posteriori probability function

$$p(i|\ell) = \frac{p(\ell|i)p(i)}{p(\ell)} \tag{3.3}$$

where the p indicates a probability density function. We will consider every intermediate value as having the same probability. $p(\ell)$ is the probability distribution of the observed power leakage, and since this value is not known directly, it can be calculated by decomposing it as a sum of the probability functions of the leakage for each of the possible intermediate values (i_n), weighted by the corresponding intermediate value probabilities. This sum is, again, simply the weighted sum of the template distributions, $\sum_n p(\ell|i_n)p(i_n)$, and the weighting terms cancel since, as aforementioned, $p(i_n) = p(i)\ \forall n$. The following expression is then obtained:

$$p(i|\ell) = \frac{p(\ell|i)p(i)}{\sum_n p(\ell|i_n)p(i_n)} = \frac{p(\ell|i)}{\sum_n p(\ell|i_n)}. \tag{3.4}$$

Notice however this is the likelihood of *one* intermediate value i that was hypothesized for a key k. In fact, there are several candidate keys k, and each key implies N intermediate values for each of the

traces, indexed as $i_{k,n}$, $n \in 1 \ldots N$. Recall that an attack is successful if the attacker can replicate the intermediate values calculated in each trace by the real device, since that would imply that the key hypothesis is correct. Then, the attacker would like to model the probability of the key, which results from the joint probability density of all the intermediate values hypotheses calculated for that key. Since each trace is an independent event, the likelihood of the key is determined by the product of the $p(i_{k,n}|\ell_n)$ for each of the intermediate values hypothesized for the key k. To apply Maximum Likelihood estimation, it is sufficient to compare which of the keys leads to the highest likelihood, as per Equation 3.5:

$$\arg\max \mathcal{L}(k; i_1 \ldots i_n) = \arg\max_k \prod_n p(i_{k,n}|\ell_n). \tag{3.5}$$

Even though the distinguisher used is different from those used in the Background section, the same DPA principles still apply for all the remaining steps of the attack.

Points of interest (POI) detection

From the previous discussion of this work, it has been established that there is a small set of interesting time points where the operation under attack leaks, and most of the trace collected during the cipher is not relevant to the attack. In fact, when using multivariate normal templates, it is massively wasteful to compute the variance-covariance matrix for all the trace points when most of these samples do not include any information relevant to the attacked operation. In these cases, it is absolutely necessary to select the relevant time samples that display the best leakage, also called *points of interest*.

If one can design some methodology to adequately quantify which time instants leak the most, not only can the attacker only process those samples — saving computational time —, but also, since those instants correspond to some operation, one can also deduce immediately which operations leak the most, and therefore choose the best intermediate calculation for the observed leakage. The process of locating the points where the circuit leaks the most is called point of interest (POI) detection. Though it is most commonly applied to locating leakage in the time domain, it can also be used for finding leakage in the spatial domain or, indeed, in any other suitable domain.

The first proposed method to achieve POI detection was the *sum of differences* method, which takes the means m calculated for the profiling of each operation, and calculates the sum of their pairwise differences [24]. An improved method was suggested by Gierlichs et al. [21], named the sum of squared differences (abbreviated as *sosd*), which replaces the pairwise differences with their square:

$$\sum_{j,\ell=1}^{K} (m_j - m_\ell)^2 \;\; \forall\, j, \ell. \tag{3.6}$$

Intuitively, one can understand that by determining the places where the mean traces tend to differ the most, these points are those that vary the most under the operations considered and should be chosen for a differential attack. When capturing at a high sampling rate, many points in the trace correspond to the same operation, which appear as clusters of points in the sum of differences trace. Since these points are redundant, one should choose the best point of each cluster.

Instead of subtracting the means, other more advanced methods have been proposed, namely the t-test. The t-test uses two classes and compares them in order to reject a null hypothesis: that the distribution of the two classes has the same mean. François Durvaux and F.X.-Standaert propose [25] a fixed vs. fixed test, where two classes with N traces with time instants τ are captured with different fixed values (i.e., two different fixed intermediate values) and the t value is calculated using the estimators of the means m and variances s^2 of the two sets:

$$t = \frac{m_1(\tau) - m_2(\tau)}{\sqrt{\frac{s_1^2(\tau)}{N_1} + \frac{s_2^2(\tau)}{N_2}}} \tag{3.7}$$

where the time instants τ corresponding to the largest t-values indicate the relevant points of interest.

Variations of the t-test exist depending on the sets chosen. For example, another alternative is the usage of a fixed vs. random test, where one of the classes has a fixed intermediate value, but the other has a random value instead. An example follows. For a device running an AES cipher, one can collect traces that either have a fixed first round SBOX output of zero (0), or a random value in that same operation. For better results, the trace capture should shuffle the traces that have fixed values with the random ones, to avoid capturing several sequences of traces with the same fixed value. For each point in the traces, the t-test is calculated, and the points with highest t-test values are those where the mean value is the most different between the random trace samples and the fixed samples, which is a clear evidence of leakage.

Chapter 4

Methodology of Analysis

Side-channel attacks exploit physical observables to extract the secrets of cryptographical implementations. As such, the success of these attacks depends on the amount of random noise present in the captured signal and on the statistical methods used to extract the correct secret. To analyze the results of these attacks and to compare strategies, it will be necessary to calculate performance metrics to characterize the relative success of each attack.

In particular, it is important to define what *success* means. The uncertainty associated with the signal variability and the statistical tools used means that two attacks using signals captured in the identical conditions do not necessarily have the same results, so it becomes important to define what it means for an attack to be successful in a systematic and reproducible manner. This work will focus on differential attacks, and as such, it will begin by defining a framework for success in the context of this type of attack.

Recall that DPA is differential due to how it compares differences between measured traces and differences between the hypotheses computed with a model of power consumption. Using statistical analysis, it can reveal a key under considerable noise and without specific knowledge about the cryptographic device.

The attack will target a specific part of the key — a partial key, or *subkey*. On AES, a subkey is typically defined as being one of the 16 bytes of the key, because operations in the AES cipher are usually executed on a byte-by-byte basis. Success will be defined for subkeys, and the success of the whole key is a natural extension of the success of its parts.

So, for each of the possible values of the subkey, the value of the chosen statistical test — the *distinguisher score* — is calculated. This allows one to create an ordered list of candidate values such that the values are placed in decreasing order of the distinguisher score. This implies the first place is the subkey value with the highest distinguisher (and thus, the best candidate) and the last place is the least likely value.

In summation, for each possible value i of the secret subkey (out of N values, for AES, $N = 2^8$), the attack outputs a tuple (d_i, p_i), where d_i is the distinguisher score and p_i is the ordinal of the position of that value in the candidate list.

One of these values, say j, is the best candidate value — that is to say, $p_j = 1$, with a tuple

$(d_j, 1), d_j \geq d_i \forall i \neq j$. In order to generalize the notion of success in an attack, one may, in some cases, not be interested in exactly which value j was the best — attacks to different devices will find different secret subkeys. Instead of indexing the values by their corresponding subkey, instead let us introduce the notation D_1, which denotes the best distinguisher score of all the values in the attack, while D_2 would denote the second best, and so on, such that $D_1 = d_j$ if $p_j = 1$. So, alternatively, one can say that the best candidate value j has a pair $(d_j = D_1, 1)$ with the highest distinguisher, where $D_1 \geq d_i \ : \ \forall \, i \neq j$.

4.1 Position analysis

One way of determining the performance of the attack is by testing the attack methodology under controlled conditions, that is to say, under a certain experimental setting, and knowing in advance which of the possible values for the subkey is the correct one. The attacker can then gather information about the performance of the attack, which can then be used to estimate the success of a real attack in similar conditions, where then the secret key will not be known.

The correct value pair will be denoted by a star, as $(d_\star, p_\star)$. In that case, it is possible to locate the correct value of the secret among the candidates listed by the attack. An immediate question follows: where did the attack place the correct value of the subkey? Is it the best hypothesis? Is it somewhere in the middle of the list? This can readily be answered by the position of the correct value in the hypotheses list, $p_\star$.

Success rate (SR)

After determining the position of the correct value $p_\star$, the most obvious course of action is to check whether the attack determined the correct value as the best candidate, that is, if $p_\star \equiv 1$. In that case, the attack to that partial key was successful. However, the success of a single attack does not adequately represent its performance and it is necessary to repeat it several times. The success rate can then be calculated as the ratio of attacks, ran under the same conditions, that yielded $p_\star = 1$:

$$SR = \tilde{P}(p_\star = 1) = \frac{\#Attacks_{p_\star=1}}{\#Attacks_{p_\star=1} + \#Attacks_{p_\star \neq 1}} = \frac{\#Attacks_{p_\star=1}}{N}. \tag{4.1}$$

The success rate gives an estimate of the probability $\tilde{P}$ of having the correct value of the subkey as the best candidate, for an attack ran under the same conditions as the ones used to estimate it [26].

Success rate of order n

The previous success rate can be generalized into an estimate of the probability of having the correct value not as the first candidate, but as the nth candidate or higher. In that sense, the success rate is equivalent to having $n = 1$ [26].

This is useful since the attacker will often not only test the best candidate, but also a few hypotheses down the list in order to check for the correct subkey. With this definition, one may be interested in

estimating the probability of having the correct value in, for example, the top-10. The calculation is similar, but instead of only selecting attacks where the correct value has position one, all attacks in which the position is lower than n are selected:

$$SR_n = \tilde{P}(p_\star \leq n) = \frac{\#Attacks_{p_\star \leq n}}{\#Attacks_{p_\star \leq n} + \#Attacks_{p_\star > n}} = \frac{\#Attacks_{p_\star \leq n}}{N}. \tag{4.2}$$

Average correct position and Guessing Entropy (GE)

The correct value position can be interpreted in a different way: if the correct value is the nth one, then the attacker would need to test all keys up to n to find the correct value. Then, the correct value position also encodes how many hypotheses can be discarded and how many of the best candidates actually need to be tested.

After collecting N attacks ran under the same conditions, one may estimate what is the mean position of the correct secret in the candidate list. Consider the notation $E(\cdot)$ as representing the expected value, also known as *expectation*. The expected correct position is then defined as the product of each position i, times the probability of the correct subkey occupying such position, or mathematically,

$$E(p_\star) = \sum_i i \cdot P(p_\star = i). \tag{4.3}$$

Then, the estimator of the expected correct position can be mathematically expressed as the *average correct position* $\tilde{E}(p_\star)$, computed as

$$\tilde{E}(p_\star) = \frac{\sum_j^N p_{(\star,j)}}{N}, \tag{4.4}$$

where j represents each attack out of N total attacks.

This metric was named by Standaert et al. [26] as the guessing entropy. The nomenclature *guessing entropy* is related to the fact that the number of positions that have to be tested to guess the correct key is, in a sense, a measure of uncertainty or missing information.

This notion is related to the information-theoretical concept of an 'equality question' [27]. Testing a key is an equality question since one is checking if the correct key is equal to the one being tested. So, in this sense, the guessing entropy is defined as the number of questions that must be made, on average, in order to guess the correct answer.

Since entropy is more often measured in bits, other authors [27] [28] usually define it as being the logarithm base 2 of the previous metric. As this is usual practice, the equation 4.4 will be called the *average correct position*, while the *guessing entropy* will be thus defined as

$$GE = \log_2(\tilde{E}(p_\star)) \text{ bits}. \tag{4.5}$$

Remaining entropy (RE) or Marginal Guesswork

Instead of the average position of the correct subkey, the attacker may be interested in a worst-case analysis — what is the worst position the correct subkey will occupy for most of the attacks? A percentile

of the correct position is defined as the position above which $a\%$ of the correct candidates will fall. Mathematically, this is expressed as the value for which the ratio of attacks with positions lower or equal to that value to the total number of attacks is equal to the percentage a, or alternatively,

$$p_{a\%} \ : \ SR_{p_{a\%}} = a \times 10^{-2}. \tag{4.6}$$

Estimating the percentiles is important since one could exclude the positions that do not contain the correct candidate for the vast majority of the attacks. For example, if one calculates percentile 99, as $p_{99\%}$, one is calculating the worst position that the correct candidate will reach for 99% of the attacks. Say $p_{99\%} = 16$. This implies that 99% of the attacks have the correct value of the key in the first 16 candidates, so if one excludes the remaining possibilities with great confidence, one only has the uncertainty of testing each of the 16 best candidates. Testing 16 candidates is in fact the same complexity as iterating over 4 bits of input, so, one can say the remaining entropy for 99% of the attacks, or $RE_{99\%}$, is four bits. More generally, this can be calculated by taking the logarithm base 2 of the quantile, that is,

$$RE_{a\%} = \log_2 (p_{a\%}) \ bits. \tag{4.7}$$

This metric has been named *Marginal Guesswork* by Köpf et al. [27]. The main difference to the Guessing Entropy is that instead of estimating the average entropy, it expresses the maximum entropy required for some defined percentage of cases.

Shannon Entropy (SE)

The attacker may be interested in calculating the entropy of the subkey taking into account the uncertainty of the position of the correct subkey accross all positions. For several attacks under the same conditions, one can plot the positions of the correct value of the subkey on the candidate list as a histogram. From this, it is possible to compute the estimated probability $\tilde{P}$ that the correct key will be placed at any of the i candidate positions, as the ratio of the number of attacks where the correct value was placed at that position to the total number of attacks ran, or mathematically,

$$\tilde{P}(p_\star = i) = \frac{\#attacks_{p_\star = i}}{\#attacks_{p_\star = i} + \#attacks_{p_\star \neq i}} = \frac{\#attacks_{p_\star = i}}{N}. \tag{4.8}$$

The estimated probability encodes the likelihood of the correct value being placed in that position. This represents uncertainty, and one can use the entropy concept from information theory, as derived originally by Claude Shannon, to compute it:

$$SE = H(p_\star) = -\sum_{i=1}^{N} \tilde{P}(p_\star = i) \log_2 (\tilde{P}(p_\star = i)) \ bits. \tag{4.9}$$

However, do not confuse this metric with the guessing entropy: while the guessing entropy represents the average number of optimal questions needed to guess the correct value, the Shannon entropy

represents the number of questions needed to know the *position* of the correct value [27] [29]. So, the Shannon entropy represents the *uncertainty* of the correct value position over all possible candidate positions.

One can think of it in a simplified example: imagine that the correct value is equally likely to be in any of 256 possible candidates. The Shannon entropy is trivally 8 bits, since the correct value can be placed in any of the 2^8 positions with the same probability. However, this is not the guessing entropy: notice that is *very* unlikely one will have to check all the 256 candidates in order to determine the correct one. Since the correct value is equally likely to be placed in any position, in some cases the correct value will be placed high on top of the candidate list, requiring very few questions, and in just as many cases the correct value will sink to the bottom of the candidate list. In essence, one can expect to only ask, on average, 128.5 questions ($\approx 2^7$) to correctly guess the value of the subkey, corresponding to a guessing entropy of 7 bits. [1]

4.2 Distinguisher analysis

Another way of evaluating success is based on the confidence of the result given by the statistical test. Different attacks can use different statistical tests in order to evaluate their confidence in each of the hypothesis. While the value of the distinguisher allows one to effectively discern the best candidate value from the set of possibilities, it does not directly inform the attacker of the confidence in the success of the attack. In order to evaluate success, one must ask: to what extent does the distinguisher *distinguish* the best candidate from the rest?

Intuitively, one understands that if the distinguisher score of the best candidate is a much higher value than any other, then the selected candidate is by far the best one, and this is likely a sign of a successful attack.

On the other hand, what if the attack's best candidate is only slightly better than the remaining candidates? The best candidate may well be, in fact, a spurious peak of the distinguisher score. In that case, since all the distinguisher scores are so similar, there is a chance that the hypothesis with the best distinguisher score is not the correct one and the attack has not been successful.

Thus, the difference of the wrong candidates' distinguisher scores to the best candidate's distinguisher score is a measure of the ability of the attack to effectively discern the correct key from the wrong ones. The term *distance* will be used to denote the difference of the best candidate key to other keys.

The objective is then to capture a measure of the *degree of separation* between the best candidate and the remaining ones using a metric based on a distance.

In Figure 4.1, two situations are depicted: a successful attack with a small number of traces on the left, and a successful attack with a large number of traces on the right side. In each plot, the correlation of each possible value is plotted, with the correct value showing a small spike on the left plot, and a much

[1] The expected number of questions for a guessing game with equiprobable probabilities and one correct value out of N is (N+1)/2. For further discussion on the guessing game entropy bounds, check the problem description as posed by Massey [29] and the Lemma 2.1 as explained by De Santis et al. [30]

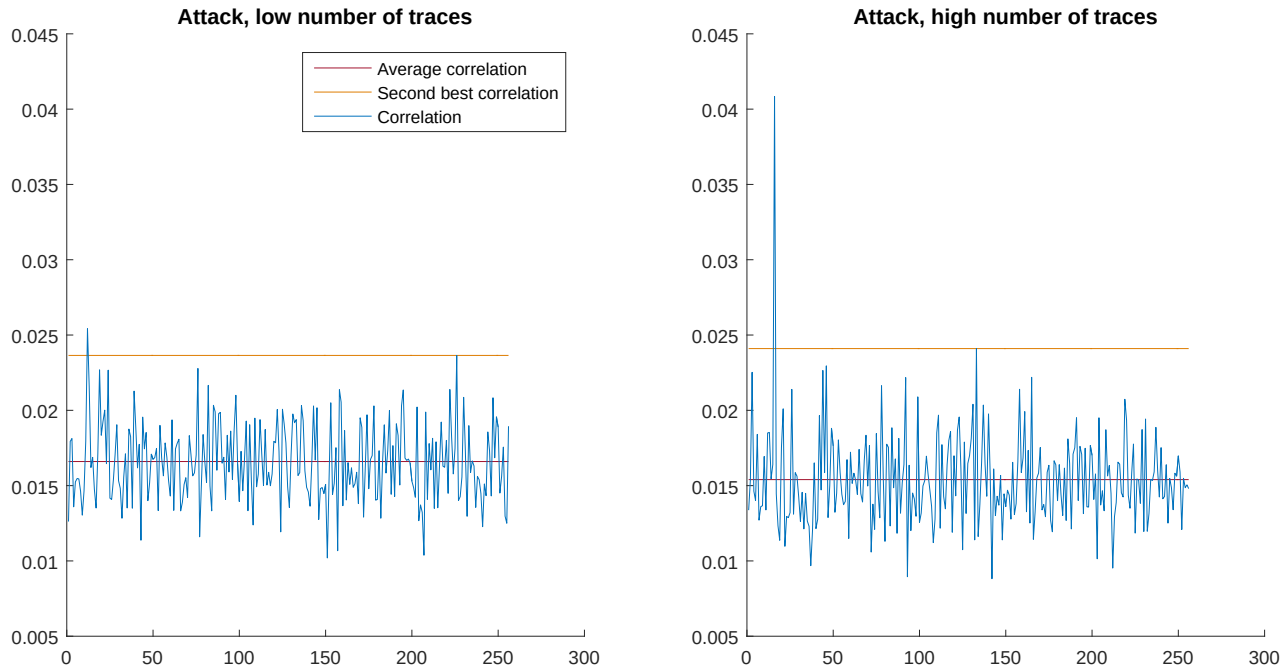

Figure 4.1: Distinguisher scores for each of the 256 possible values of an 8-bit subkey in a correlation attack (x-axis: possible values of the subkey, left: attack with small set of traces, right: attack with an expanded set of traces, including those used in the other attack; both attacks performed in same experimental conditions).

more noticeable one on the right one (the exact value of the candidate for the subkey is not relevant, however, a careful eye will notice that the large spike on the right does indeed have a corresponding smaller spike for the same value in the left plot).

Readily, one can conclude that the attack with a larger amount of traces has a much more significant correlation for the correct value than the attack with a lower number of traces. How can one translate this visual assertion into a mathematical construction?

Distance to the average

A first approximation may be to calculate the distance ℓ' between the best candidate and the average of the remaining $N-1$ keys (depicted in Figure 4.1 as a red line in both cases).

$$\ell' = D_1 - \frac{\sum_{j=2}^{N}(D_j)}{N-1} \tag{4.10}$$

While taking the average of the remaining subkeys is a reasonable approach, in weak attacks there are several spike values of the remaining candidate subkeys that stray considerably from their average, albeit to a limited extent. For example, in the attack depicted on the left, in Figure 4.1, there are several values that are almost just as distant from the average as the best candidate. This would mean that, for this metric, distances up to a certain value (the upper bound of noise) would not imply any real separation between the best and remaining candidates. Worse still, the variance of the distinguisher noise depends on the attack conditions, so such a reference value would necessarily differ between attacks.

Distance to the second candidate

Consider instead to compute the distance between the best candidate and the second best candidate (yellow line in the plots of Figure 4.1).

$$\ell = D_1 - D_2 \tag{4.11}$$

This approach immediately encodes the distance between the best candidate and the rest of the pack. While the second best candidates will differ in every attack, it will always be the best competing hypothesis, and a greater distance to the second hypothesis is a good indicator of the performance of the attack. Because the random peaks of wrong hypotheses tend to stray from the average to a limited extent, the second best candidate is a good upper bound of the noise present in the distinguisher. This is illustrated by the negligible variation in the yellow line in Figure 4.1: although the attacks have different second best candidates, they both have similar peak values for their respective distinguisher.

The distance to the second candidate, as it is merely a difference between two distinguisher scores, is dependent on the type of distinguisher. For example, recall that template attacks do not typically employ correlation, but instead use a strategy based on the maximization of likelihood. So, in the case of template attacks, the distance between the best candidate and the second best candidate is a difference between two likelihoods, and since likelihoods are unbounded and vary considerably depending on the input, this difference is no longer comparable between different attacks: perhaps what is a large distance in one attack can be a very small one in other, if the values of the latter are of a much higher order.

Other solutions are possible to deal with the variance of the distinguisher noise. Instead of taking the distance to the second best candidate, it is possible to take the distance to an average of the runner-up candidates, for example, the five best candidates after the first. These metrics however compute a similar upper bound for noise, and thus produce very similar results at the expense of more calculations.

Normalized distance

In order to properly account for unbounded distinguishers, consider instead the *normalized distance*, w. The normalized distance is the same as the distance between the two best candidates, but normalized by the second candidate. As such, a normalized distance of 0.3 means that the best candidate has a distinguisher score 30% higher than the second best candidate. Notice that with this metric, one can effectively characterize the degree of separation between the best candidate and the remaining ones, in a manner that is completely independent of the kind of distinguisher used: the normalized distance refers to a degree of separation whether it is applied to correlation, likelihood or a probability.

$$w = \frac{D_1 - D_2}{D_2} = \frac{D_1}{D_2} - 1 \tag{4.12}$$

Care must be taken however since not all types of distinguishers have the same statistical power. While this metric is extremely useful to compare attacks using the same type of distinguisher (correlation vs. correlation, likelihood vs. likelihood, ...), different statistical tests have different sensitivities (which

this metric makes evident). This means that if one has two attacks with different types of distinguishers, e.g. correlation and likelihood, with the same normalized distance — and thus the same degree of separation between the best key and the remaining keys — the attack using the likelihood test will not necessarily show the same success rate as the correlation test, because the degree of separation has a different importance for each of the statistical tests used. In conclusion, for different types of distinguisher, while the normalized distance means the *same*, it must be interpreted *differently*.

Correct normalized distance

When evaluating the success of the attack, if the correct subkey is known, it may be useful to compute its *correct* normalized distance $w_\star$ from the second best candidate. The notation $d_\star$ denotes the distinguisher score for the known correct value of the partial key (or subkey).

$$w_\star = \frac{d_\star - D_2}{D_2} = \frac{d_\star}{D_2} - 1 \tag{4.13}$$

In this case, negative values of the correct normalized distance mean that the correct value is not classified as the best candidate by the attack, while positive values indicate the correct value is the best candidate, since it has a positive distance above the second best candidate.

A similar metric, the distance between the correct and best wrong candidate normalized by the standard deviation of the candidates' distinguishers, is presented in Whitnall and Oswald. [31]. The authors named it *Nearest-rival distinguishing power*. Such a measure is effectively independent of the type of distinguisher used, which addresses the aforementioned critic on non-normalized distances.

4.3 From partial keys to complete keys

One can only achieve a successful decryption of a symmetric cipher by revealing the entire key. If these attacks use a divide-and-conquer strategy, it is then necessary to, first, define the strategies that yield the best result to rebuild the key and, finally, define metrics that evaluate success on the recovery of the entire key in terms of the success of the partial key recoveries. In this section, rebuilding strategies will be presented, including an optimal strategy to build the complete keys from the partial key.

These strategies dictate how to construct the ordered list of best guesses, but it is not expected that the attacker will in fact build the entire list — doing so can be computationally demanding, depending on the key length. Instead, the attacker will progressively generate each guess as defined by some strategy, stopping after finding the correct key or completing a reasonable number of trials without success.

Success rate for a complete key

Whichever strategy used, the best guess for the complete key is trivially the combination of the best guesses for each of the partial keys. The success rate for a complete key (SR^{CK}) is redefined as the ratio of attacks (out of N total attacks) that simultaneously reported all the correct $i = 1, ..., K$ subkeys in the first position:

$$SR^{CK} = \frac{\#Attacks_{p_{i,\star} \leq 1, \forall i=1,\ldots,K}}{N}. \tag{4.14}$$

This is equivalent to the joint success rate of every partial key SR^i. That is to say, the probability that the best guess of the complete key is the the correct key, SR^{CK}, is simply the product of the probabilities that our best guesses for the K partial keys are correct, or mathematically,

$$SR^{CK} = \prod_{i=1}^{K} SR^i \tag{4.15}$$

In practice, this metric may be enough for most attackers. Instead of trying to guess the correct key from a single attack by iterating optimal guesses, a simpler alternative is to attack subsets of the collected traces in several runs and check the top guesses for the subkeys. By executing several attacks, even for a small success rate, the correct key is revealed due to the fact that the small number of correct hypotheses will be repeated between attacks, while the wrong hypotheses will vary considerably. This is the principle behind the strategy described in Thillard et al. [32].

However, the objective is to benchmark each attack, not a sequence of attacks that may confirm each other's hypotheses, even if such practices are more common than extracting multiple guesses from a single attack. As such, key reassembly strategies will be presented in the context of the results of single attacks.

Iterative bruteforcing strategy

After testing the best guess for the complete key, which guesses should follow? One of the simplest strategies is the brute-forcing of the partial keys with the weakest confidence. Starting with the first candidates for each partial key, take the weakest partial key and test all the remaining values. If the key hasn't yet been found, continue testing by also iterating over the value of the second weakest key. This process continues, sequentially brute-forcing each subkey in order of least to highest confidence, as shown in Table 4.1.

Table 4.1: Candidate list for a weakest partial key brute-force strategy with 16 partial keys ordered by confidence. Each partial key has a list of 256 possible values, their positions in the list are shown in the table. The iteration of the weakest key is shown in red, the second weakest in orange.

p_{16}	p_{15}	...	p_2 (second least confident)	p_1 (least confident)
1	1	...	1	1
1	1	...	1	2
1	1	...	1	...
1	1	...	1	256
1	1	...	2	1
1	1	...	2	...
1	1	...	2	256
1	1	...	...	...
1	1	...	256	256
...	...	...	...	...

For this strategy to be possible, the attacker has to characterize the results of each partial key by

using one of the aforementioned strategies. For example, to establish the confidence of the results of each subkey, one of the metrics that can be used is the average correct position for each partial key. In that case, the first subkey to be iterated is the one with the highest average correct position, which implies that the correct value will be situated further down the list.

In particular, using the brute-force strategy, it is possible to define the average position of the complete key $\tilde{E}(p_\star^{CK-BF})$, in terms of the average position of the partial keys $\tilde{E}(p_\star^i)$, for every i subkey ordered by increasing confidence. The reasoning is as follows: take the first candidates for each partial key, that is your first hypothesis for the complete key. To that one (1) hypothesis, sum the expected number of guesses for the weakest partial key (minus one of the values, which was already tested as the first candidate), that is to say, the average position of the weakest partial key minus 1. Mathematically, so far, this is $1 + (\tilde{E}(p_\star^1) - 1)$.

To continue, brute-force the second weakest partial key ($i = 2$), but now consider that, when brute-forcing the second weakest subkey, the weakest subkey also has to be iterated. The combinations needed are multiplied by the total number of values in the weakest subkey, N. As such, to proceed, each of the average correct positions is multiplied by the corresponding power of the total number of values and added to the sum. In conclusion, the estimated expectation for the position of the correct key using the brute-force strategy, is defined from the estimators for the K partial keys as

$$\tilde{E}(p_\star^{CK-BF}) = 1 + \sum_{i=1}^{K} N^{(i-1)}(\tilde{E}(p_\star^i) - 1), \tag{4.16}$$

where the partial keys being tested (denoted as i) are ordered by increasing confidence.

As before, this expectation encodes the guessing entropy for the brute-force strategy, computed by taking the logarithm base 2 of the result:

$$GE^{CK-BF} = \log_2(\tilde{E}(p_\star^{CK-BF})). \tag{4.17}$$

The brute-force strategy is not optimal, but it is simple to implement and allows the attacker to calculate an upper bound of complete key success based on the performance of the partial key attacks. This may enable the attacker to readily compare the performance of several attack techniques even if the brute-force strategy is not used. What follows are other strategies which can improve on the number of attempts to find the best key and the memory requirements to do so.

Percentile-limited strategy

Considering the brute-force strategy, it may become apparent that testing all the hypotheses for the weak subkeys may not be optimal. Even in the candidate lists of the weakest subkeys, the bottom positions may have a negligible probability of being correct. As such, instead of iterating over all positions in each subkey, the attacker may choose to iterate only over the positions that are likely to contain the correct partial key.

In that case, the attacker can leverage the percentile of the correct position, $p_{a\%}$, to define the cov-

erage of hypotheses. Since the probability of the complete key is the joint probability of partial keys, the coverage of $a\%$ of the correct positions for each of the K partial keys is equivalent to the coverage of $100 \times (a \times 10^{-2})^K$ % of the positions for the complete key. That is to say, if one limits the brute-force attack to a 99%-percentile of the candidates for each of 16 partial keys, one expects to cover $(0.99)^{16} \approx 0.85 \rightarrow 85\%$ of all correct complete keys.

As in the previous strategy, it is possible to calculate, from the partial keys, the expected position for the percentile strategy. The reasoning is the same, but replacing the average correct position by the percentile of each subkey; and the power of N, representing the brute-forced hypotheses, by the product of the percentiles of the previous subkeys. Mathematically,

$$\tilde{E}(p_{(a)^K}^{CK-PC}) = 1 + \sum_{i=1}^{K} (p_{a\%}^{i} - 1) \prod_{j=1}^{i-1} p_{a\%}^{j} \ , \tag{4.18}$$

taking care to order the i and j subkeys in increasing order of confidence.

Optimal rebuilding strategy

When executing an attack, the attack outputs the ordered list of likelihoods for all the possible values of every subkey. In fact, the attacker is interested in the list of likelihoods for complete keys. To build the ordered list of complete keys, define the likelihood of each complete key as the joint likelihood of the partial keys, or in other words, the product of every partial key value's distinguisher scores yields a new score for the key composed of those values.

A simplified example follows. The likelihood of a complete key $CK = \{4, 9, 245\}$, composed of the 3 partial keys, $pk_1 = 4, pk_2 = 9, pk_3 = 245$, is simply the product of the their distinguishers

$$d(CK = \{4, 9, 245\}) = d_4^{(pk_1)} \times d_9^{(pk_2)} \times d_{245}^{(pk_3)}. \tag{4.19}$$

If the attacker produces, sequentially, a list ordered by the likelihood of each complete key hypothesis, that list follows the optimal order to rebuild the key with the minimal amount of guesses.

This strategy naturally depends on the values of the distinguishers in every partial key. In different instances of the attacks, these values change. As a result, it is not trivial to estimate the performance of this strategy from metrics of the partial keys taken from several attacks. The attacker may instead resort to the metrics derived from the previous strategies to compare different attack techniques.

4.4 Sample size of analysis

Some of the benchmarks proposed require running several attack retries to characterize their performance. For example, to characterize the success rate of some attack, it is necessary to run it several times against different sets of traces captured under the same conditions. This begs the question: *How many trials for each attack should one run?*

In order to obtain reproducible results, it is necessary to repeat the attack enough times to reach statistical significance, so that the values obtained in the metrics accurately represent the range of possible results that the attack can yield. In effect, one can always improve the representativeness of the metrics by simply increasing the number of repetitions, but time constraints dictate the limitation of the number of runs to a reasonable, but representative, number of attacks.

To better grasp how many attack repetitions are needed, perhaps it is best to first ask how many different attacks are possible under the same conditions. Trivially, there are endless possible results for some attack technique, depending on the noise variation in each trace or the plaintexts used to generate the traces. For perspective, in a 128-bit cipher, even just storing the possible combinations of plaintext input ($2^{128} \approx 10^{38}$) would not be possible with current storage technology. The problem of finding the number of repetitions is called a *sample size* problem.

From a review of literature, the sample size of 100 was determined to be the most common, but variation between 20 and 240 was observed [31][33][34][35]. The margin of error is estimated to be approximately 10% (for 95% of the time), as calculated by the sample size formula derived from the statistics field[2]. In any case, experimental results in literature show values to be significantly more consistent than the margin of error calculated, which implies such a margin is in fact conservative. After testing with as many as 400 retrials and verifying the true variation of the results, the sample size selected was 50, due to time constraints producing the experiments and the available equipment. In fact, experiments with significant signal processing can take several days to complete for a larger number of retrials, and the 400 retrials experiment took more than a week to complete using available hardware.

To conduct the experiments, the number of traces needed to execute each trial is selected at random from a larger set of traces, all captured during the same capture session, under the same experimental conditions. The larger set of traces is continually shuffled in order to ensure that the traces selected for each trial are randomly picked from the large set. Finally, after running 50 retrials of the same attack using the same number of traces, it will be possible to benchmark the performance of the attack against competing techniques. The metrics are presented by their average over the retrials, and an error term corresponding to the standard deviation.

Consider the subkey metrics: when extracting the metrics for each experiment, it may not be reasonable to present values for each of the partial attacks. As such, for subkey metrics, the average across the subkeys is typically calculated. In that case, such an average is typically a *global average*, denoted m, since it is taken considering every trial of each subkey as part of the same attack.

In order to characterize the error of such results, the standard deviation s is calculated. In the case of subkey metrics, the same approach is followed: the trials of all the different subkeys are pooled and the standard deviation is computed on them.

Results are presented with their average m and an error term e, called the *standard error*, which characterizes the uncertainty of the average value for one standard deviation of the sampling distribution (approximately 68%). The standard error for the average is calculated using the standard deviation of

[2]For more details on the calculation of a sample size and the corresponding margin of error, the reader is referred to introductory literature in the field of Statistics. This calculation is for a 95% confidence level and an implied percentage of success of 50%, which is effectively the worst-case scenario of maximum uncertainty.

the results s as follows:

$$e = \frac{s}{\sqrt{N}}, \tag{4.20}$$

where N is the number of trials which were used to compute the sample mean. For example, there are 16 subkeys in an AES attack, so, when 50 complete trials are run, $N = 16 \times 50 = 800$ partial trials are executed across all the subkeys. However, if a metric is calculated on the complete key then there are only $N = 50$ trials.

A plus or minus sign is used to indicate the error term. The resulting notation is then:

$$m \pm e. \tag{4.21}$$

In conclusion, the framework for analyzing every attack has been defined in order to ensure consistent and repeatable results, which can be used to compare different techniques. In the next two chapters, experimental work will be reported and the attack algorithms will be evaluated using precisely this same framework, starting with power attacks and, later, electromagnetic attacks.

Chapter 5

Power Attacks

In the previous chapters, the background concepts in the cryptological domain and the foundations of side-channel attacks were laid out, followed by a thorough description of a framework for analysis. The next step is to carry out a power attack. This choice is motivated by the fact that existing previous work [36] and infrastructure at the research facility provides a suitable entry point to achieve the objectives of this dissertation. Power signals are easier to acquire and the expected experimental results are well studied in the cryptanalytical field. This will provide the practical know-how needed in side-channel attacks to validate the techniques which will then be used in electromagnetic attacks.

By validating these techniques first on power attacks, it is possible to address the challenges of electromagnetic attacks with confidence on the robustness of the methods applied, and apply tools of the signal processing field to overcome them. After performing successful attacks using power analysis, a move to electromagnetic attacks will provide a new opportunity to test signal processing techniques as deemed adequate, in accordance to the previous results. To carry out the experimental work, an appropriate experimental setup is required to capture and process the signals of interest. A description of the setup follows in the next section.

5.1 Experimental setup

The experimental setup is comprised of three main components: the *device under attack*, which executes the targeted cryptographic algorithm with a hidden key, the *attacker device*, responsible for executing the computations needed to reveal the correct key and coordinating the attack, and the *signal acquisition device* that samples the electrical signals in its input and generates a digital signal that can be processed at the attack device. The entire process flow is represented in Figure 5.1.

First of all, the signal acquisition device has to be chosen, which must interface with a computer to enable the execution of said attacks. A personal computer (PC) digital oscilloscope is the right choice for this purpose. The oscilloscope samples the signals at the input ports and outputs the data stream via an universal serial bus (USB) interface to the computer.

There are two possible devices under attack (DUA). There is a reference board, the SAKURA-G

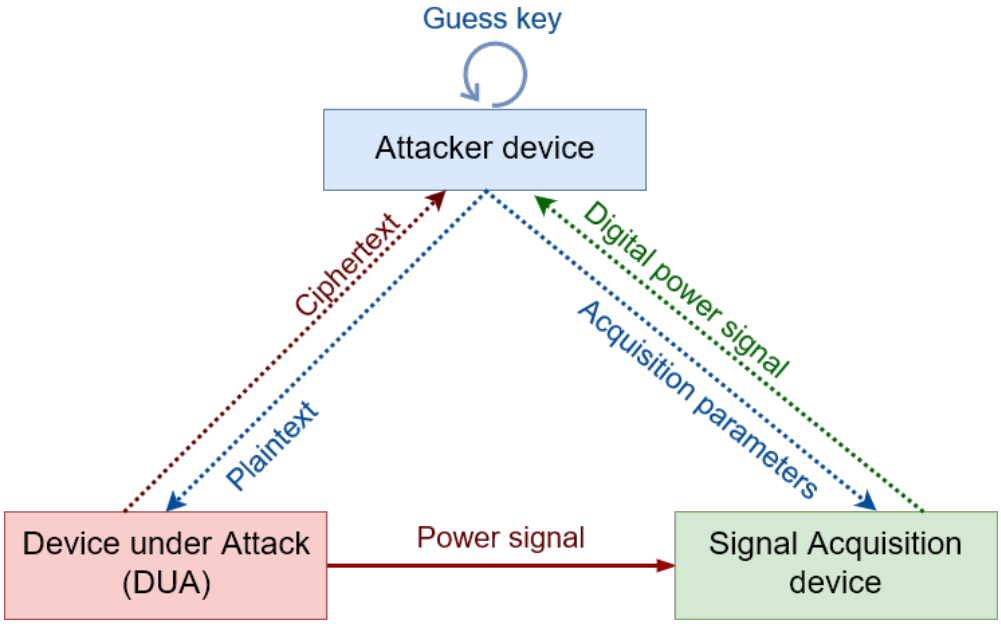

Figure 5.1: Conceptual diagram of a side-channel attack.

R1 [37], a circuit which includes a chip that executes the cryptographic routines, and components that facilitate performing the attacks and allow for the execution of the experimental work in repeatable and well-standardized conditions. The board also includes a power consumption signal output port that allows for simple execution of power attacks.

The second DUA used in this work is the smartcard. The smartcard is a programmable portable chip that can execute cryptographic routines. As before, an expansion board is used to facilitate the execution of the attacks under consistent conditions. The expansion board includes a power consumption output port, connected to the smartcard, which can be used to perform the smartcard attacks.

Finally, the attacker device is a personal computer. The capture process, the attacks and signal processing techniques will all be implemented by the computer.

For the collection of traces, the computer interfaces with the oscilloscope and the device under attack to collect the attack data. In the case of the oscilloscope, the computer defines the signal acquisition parameters and collects the resulting power traces. For the DUA, instructions are sent to encrypt a plaintext, and the resulting ciphertext is retrieved. The computer then associates the plaintext and ciphertext with the power trace that led to their calculation.

The computer uses the collected data to execute the attack, using attack scripts that attempt to derive the most likely key. In order to properly evaluate the attack methodologies, statistical data processing techniques will be implemented in the computer to extract performance metrics and produce plots for analysis of the several attack strategies used.

In conclusion, the conceptual framework of side-channel attacks was discussed, providing a brief overview of the overall experimental configuration. The specifics of the hardware and software used for the experimental work will be discussed in the next sections.

5.1.1 Hardware used

In this section, the complete setup for attacking each of the DUAs is presented with all the required equipment. The equipment will be documented, along with their characteristics and a discussion of their limitations and suitability. First and foremost, a discussion of the common equipment follows, that is, equipment used in any of the devices under attack. Later, the signal acquisition equipment will be detailed. Finally, the details of each of the devices under attack are discussed, as well as the setup that enables their attacks.

Common equipment

Executing attacks requires manipulating vast amounts of data, up to thousands of traces containing hundreds of thousands of samples. To enable the computation of the attacks, the computer used was equipped with 32 gigabits of memory and a quad-core central processing unit. Even though this is a considerable amount of memory, it may not be enough for some difficult attacks requiring more than a hundred thousand traces.

In power analysis attacks, it is of the utmost importance to have a stable power source. This avoids introducing unpredictability in the experimental results. As such, all devices under attack must be powered by an adequate laboratory bench power supply, which isolates the voltage input of the devices from unpredictable fluctuations and spurious noise coming from the electrical grid. For powering the devices under attack with stable power, a GW GPC-3020 laboratory bench power supply is used for this purpose.

For connecting the equipment, the MiniCircuits 086-2SM+ (rated from DC to 18GHz) coaxial cables are used. They feature an SMA connector (SubMiniature A), a commonly used connector for 50 Ω coaxial cables. These cables far exceed the frequencies that are relevant for the attacks and their low attenuation makes them suitable for our setup.

An amplifier is useful in order to better condition the signal before acquisition, since it may require further amplification depending on the setup used. When used, it is powered by the same power source as the device under attack, to maintain the isolation from the power grid. A low noise amplifier is available, the Mini-Circuits ZFL-1000LN+, providing a minimum gain of 20 dB (corresponding to 10 times amplification) and a typical gain of 24 dB when powered at 15 V. It features a typical noise figure of 2.9 dB, that is to say, that the relation between the signal and the noise (called SNR) is reduced by that amount after the amplification, when compared to the input SNR.

To use the amplifier, it is necessary to guarantee that no DC current is flowing in the cables. An INMET 8037 DC block is used in the laboratory when connecting a power signal output to the amplifier. It is rated up to 18 GHz, well beyond the frequencies of interest.

Signal acquisition

As aforementioned, a PC oscilloscope will be used, in this case, the PicoScope 6404D. This oscilloscope is rated with 500 MHz (-3 dB) bandwidth, which means that, for signals that vary 5×10^8 times per second, the power of the captured signal is only half of the actual channel input (-3 dB corresponds

to a $\sqrt{2}$ amplitude factor, or a 2 power factor). The oscilloscope is not suitable for signals with higher frequencies, but recall that the switching activity of the DUA is fairly low-frequency, as explained in the context of electromagnetic attacks in Section 2.2, and as such this device is well suited for our attacks.

The sampling frequency is 5 gigasamples per second (5 GHz), with up to 8 bits per sample (the amplitude resolution). There are several predetermined voltage intervals where the captured signal can be acquired, the smallest of these ranges divides the 8 bits of amplitude between -50 mV and 50 mV. Because these capture ranges are fixed, amplification adjustments may be required to ensure the signal can occupy the entire range, so as to ensure the captured signal adequately uses the available 8 bits.

The oscilloscope supports up to 4 signal channels and one extra trigger channel. One of the channels is to be connected to the device's power consumption signal, to be digitized. The trigger signal can either be connected to the dedicated trigger channel or one of the remaining signal channels.

Attacking the reference board

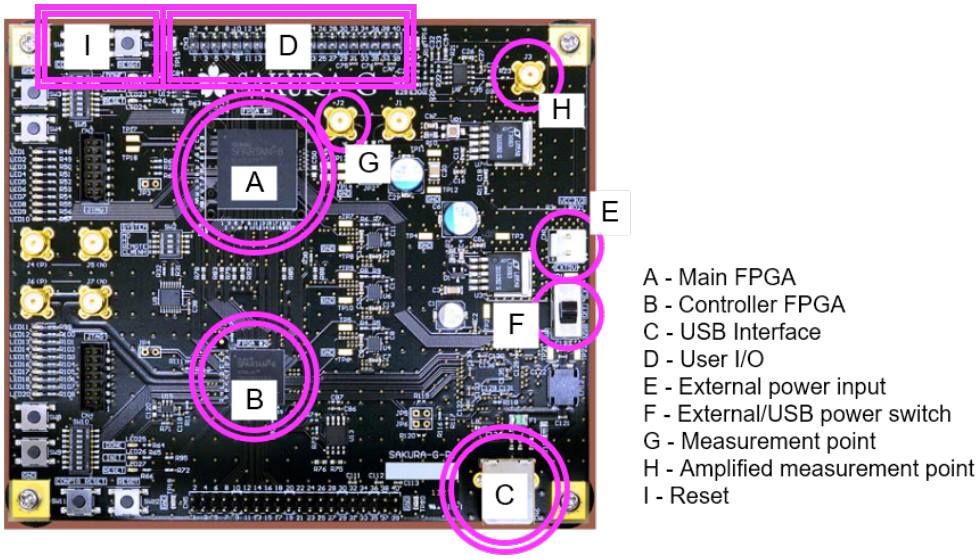

Figure 5.2: Annotated picture of the SAKURA-G. Single-line shapes have labels outside to allow the reader to see the smaller component in question. SAKURA board picture sourced from [37].

First, let's consider the power attacks directed at the reference board. The SAKURA-G reference board includes two main processing components: one where the cryptographic routines are executed, which will be the target of the attack, and a second one that interfaces with it, enabling fine-grained control and monitoring. These two components are implemented using field-programmable gate arrays (FPGAs), which are integrated circuits that can be configured after being manufactured. The use of FPGAs provides extra flexibility, since it allows for the configuration of different algorithms to be attacked and the customization of the physical implementation, simply by modifying the implemented circuit.

The main chip also generates trigger signals for the oscilloscope, output via the User I/O interface for the main FPGA (labelled as D in Figure 5.2). These facilitate consistent capture of time-aligned traces using an oscilloscope. Several trigger signals are available, depending on the programming of the main chip, such as a trigger for the first round, the last round, or the entire execution of the algorithm.

The controller chip on the board coordinates the communication of the operations to the main chip. The board is connected to a computer via an USB interface where it receives the commands for the controller specifying the operations to execute, and relays responses back to the computer. The USB cord used includes a magnetic choke in the middle to avoid contaminating the reference board with computer-originated noise.

The board provides a built-in 1 Ω resistor in series with the main FPGA power supply, and a corresponding measuring point (as depicted in G, Figure 5.2) used to capture the power consumption of the main chip, along with a 20 dB amplified port (H, Figure 5.2) that facilitates the acquisition of power traces. These ports, implemented as SMA connectors, can be used directly in order to execute power attacks.

Having detailed the equipment under attack, it must then be configured for testing. A diagram of the complete setup for attacking the reference board is pictured in Figure 5.3.

One begins by setting up the power supply for a 5 V output voltage and connecting it to the reference board (via E in Figure 5.2). The board is configured to use the external power source by setting a switch (labelled as F in the same figure). The attacker then chooses one of the available signal sources, either amplified or not, and connects it to an available channel in the oscilloscope. The appropriate trigger signal is chosen from the I/O pins (labelled as D) and connected to the oscilloscope. Finally, the board and oscilloscope are connected to the computer (via port C in the reference board) and the equipment is now ready to begin an attack.

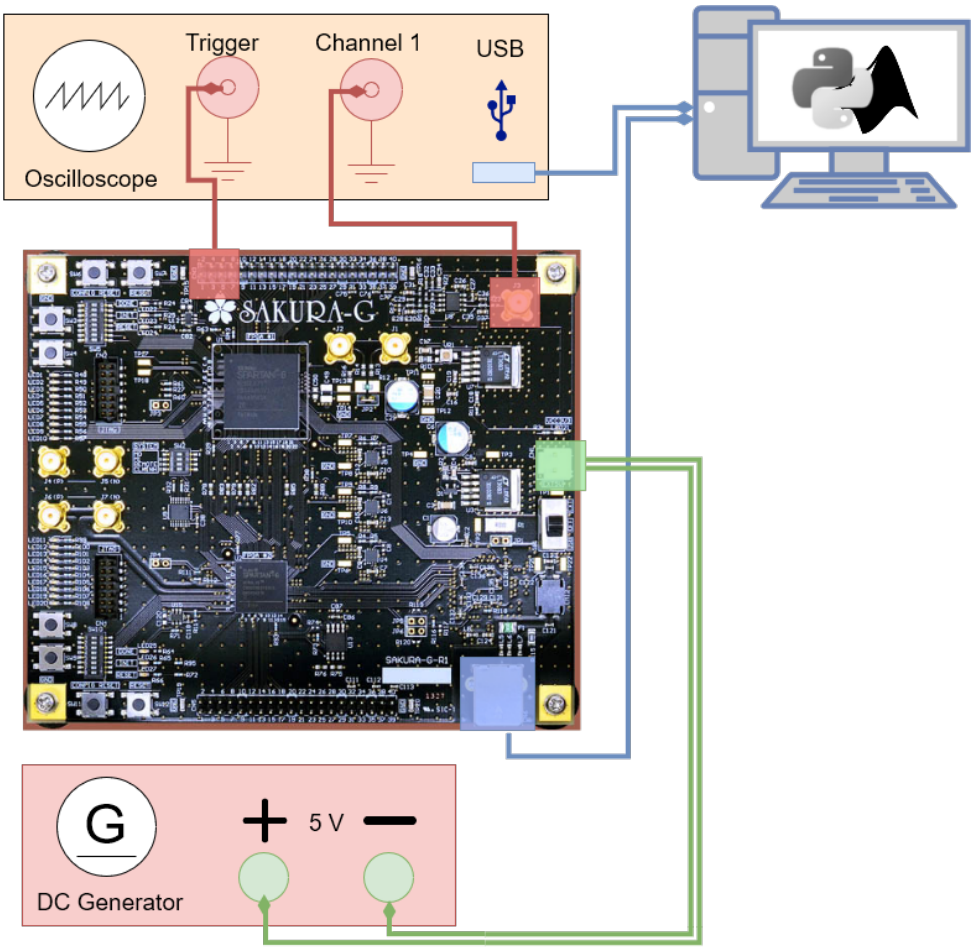

Figure 5.3: Diagram of the complete setup for the SAKURA G reference board power attack.

Attacking the smartcard

The smartcard is a plastic card that includes an embedded microcontroller, the Atmel AVR ATMega163, and a set of metal contacts used for powering the microcontroller and exchanging data. The microcontroller includes built-in memory which allows for the programming of embedded software, retained even when the smartcard is disconnected. The microcontroller is responsible for processing the serial requests, computing the requested cryptographic routines and returning the result.

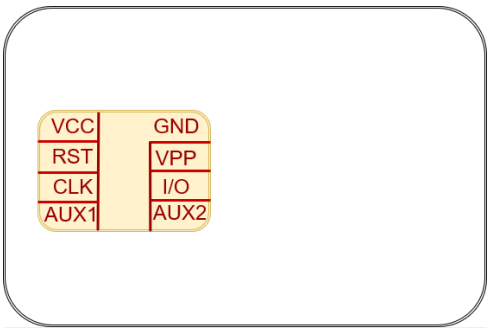

Figure 5.4: Diagram of a smartcard, annotated with the name of each metal contact.

In particular, the nine contacts, represented in Figure 5.4, are:

— VCC contact, used as the power supply input;

— GND contact, for the reference voltage input;

— RST contact, used to trigger a microcontroller reset;

— CLK contact, used to provide a timing signal (optional, since the smartcard may have its own internal clock);

— VPP contact, which is the programming voltage input;

— the auxiliary contacts, AUX1 and AUX2, with usage dependent on the embedded software.

For the smartcard attacks, an expansion board, called SAKURA-W [38], is placed on top of the SAKURA-G R2 reference board. This expansion board includes a smartcard slot and maps the smartcard pins to the I/O of the board.

When placing the expansion board on top of the reference board, the reference board is configured so as to disable its own cryptographic module and use instead the smartcard for all the cryptographic routines. When in use, the cryptographic routines are executed in the smartcard microcontroller instead of the main FPGA, while still being commanded by the same controller chip, now reprogrammed to interface with the smartcard.

As in the reference board case, the expansion board provides a built-in 1 Ω resistor in series with the power source of the smartcard, both in the power supply side (VCC) and the reference voltage side (GND). Both ports can be used to carry out power attacks. Unlike the reference board, an amplified port is not present in the expansion board.

The final setup for the smartcard attack is very similar to the one used for the FPGA attack. The diagram of the complete setup is provided in Figure 5.5.

As before, the power supply must be set for a 5 V output voltage, and the board must be configured to use the external power source by setting the appropriate switch. The attacker then proceeds to choose one of the two available signal sources, and connects it to an oscilloscope channel. The trigger signal is selected among the I/O pins and connected to the oscilloscope — SAKURA W maps the auxiliary contacts of the smartcard, used for triggering, to the pins for easy access. After the board and oscilloscope are connected to the computer, the equipment is finally ready to begin a smartcard attack.

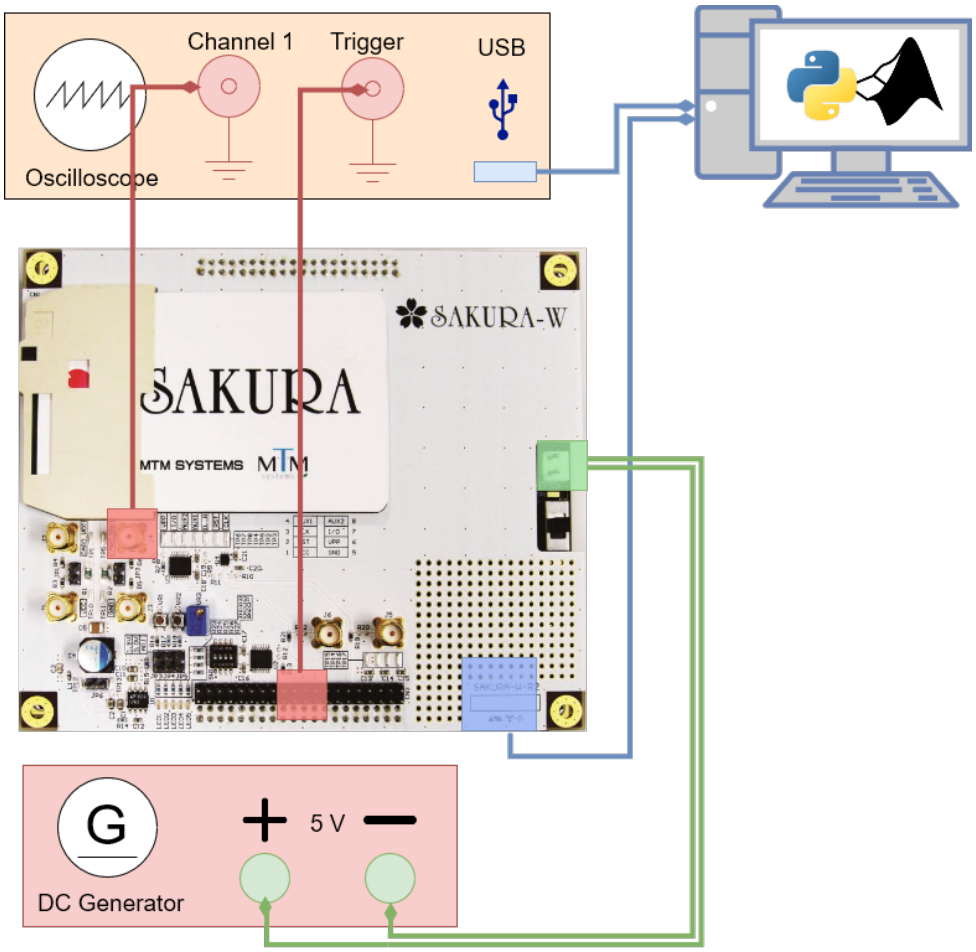

Figure 5.5: Diagram of the complete setup for the smartcard power attack.

5.1.2 Software developed

The software is what enables the execution of cryptographic routines, and the capture and computation of the attacks. The software stack used was based on the existing work by João Lopes [36]. There are two main types of software used: the software implemented on the attack devices, called *embedded software*, and the *offensive software*, used for carrying out the attacks.

Regarding the offensive software, it is comprised of two essential types of modules: the trace capture module and the attack modules. On one hand, the trace capture module is responsible for the communication with the oscilloscope and the device under attack, so as to coordinate the operation of the cipher in tandem with the signal capture of the oscilloscope. It is this orchestration between the two devices that the module must perform, ensuring all the traces are time-aligned and captured under the same conditions. On the other hand, the attack modules are comprised of implementations of the several attack techniques already discussed, using the data gathered from the trace capture module.

In the devices, the embedded software encodes the implementation of the cipher in study and the communication protocol to interact with the computer. To facilitate the attacks, the implementation may include a signal that indicates the processing of the cipher, which can be used to trigger the oscilloscope and to facilitate time-alignment of the captured traces.

Trace capture module

The trace capture module was first developed by João Lopes [36]. The module is initialized with the intended characteristics for the simulation of the attack, namely, what secret key the device will hide and which source to use for the plaintexts in the attack.

The module must first establish initial communication with the oscilloscope to define the sampling conditions, such as the bit depth and sample frequency. These ensure the capture process is systematic and consistent across all the traces in the session, as defined before starting any capture.

The module proceeds to prepare the device under attack by establishing a communication channel with it and setting the key that will be used, making sure it is ready to start ciphering. Having both the oscilloscope and the device ready, the module begins a loop that will capture the execution of the device for the number of traces requested at the start.

In this loop, first the oscilloscope's trigger is armed, so as to start the capture as soon as the execution is signaled by the device, then the module provides a plaintext to the target device and requests it to execute the cipher. Finally, at the end of the operation, the module collects both the resulting ciphertext from the device and the resulting trace from the oscilloscope.

In order to cope with the different types of communication required, the Python language was chosen due to its flexibility and, most importantly, the availability of the *Pico-Python* library which enables communication with the oscilloscope, developed by Colin O'Flynn and Mark Harfouche. Python also provides a serial communication library that is used for communication with the smartcard, PySerial, developed by Chris Liechti. For communication with the FPGA encryption core, the module invokes an external, custom-made, compiled executable, also developed by João Lopes [36].

To better support the objectives of this work, the existing code was refactored. In particular, the capture module developed by João Lopes is in fact best described as a collection of several scripts tuned for each attack scenario. By developing a more abstract code base, it was possible to execute captures with small variations of parameters using a single, coherent Python script.

The refactoring work started by separating each of the communication interfaces into small libraries that are called by the main script. Using an object-oriented approach, the Oscilloscope object was defined and all the operations that could be performed in it. This approach also allowed the device under test to be abstracted, in order for the same script to be able to communicate with a smartcard or a FPGA, running otherwise the same attack. The final script builds a Capture object, which manipulates the Oscilloscope object and communicates with the target device. This can be defined to be either the FPGA or the smartcard, in the latter case, using one of two libraries coded for each of the available smartcard readers — SAKURA-W, a smartcard-reading addon for the SAKURA platform; or the Infinity USB Unlimited card programmer, which can also be used for communicating with smartcards.

Regarding the sampling parameters, which are provided to define the configuration of the acquisition, standardization was carried out in order to better align the script with common practices in the instrumentation and signal processing field. As an example, the calculation of the number of samples to capture in a trace was previously an intricate calculation that used ratios of the sampling period and the duration, this was replaced by the more usual measure of the sampling frequency.

Several features provided by the oscilloscope were newly introduced in the script, such as the detection of overflow for captured values outside the oscilloscope range, the possibility of shifting the signal using the DC offset function, the ability to retain samples captured in memory before the capture was triggered, called *pretrigger*, and the possibility of turning off channels. While this last feature may sound trivial, due to the way the signal acquisition is carried out in the oscilloscope, the maximum sampling frequency is only attainable if only one channel is enabled. As such, to allow the user to always be able to set the maximum frequency allowed by their configuration, all channels that are not configured are now, by default, explicitly turned off by the script.

As aforementioned, the previous existing capture scripts were customized for different attack scenarios, including commented code to switch options. For example, one may be interested in feeding the device with a constant plaintext for all attacks, or perhaps to iterate the plaintexts stored in some file, or even just using a randomly generated plaintext for every trace. An equivalent decision can be made regarding the value of the key: it can be fixed, random or iterated from a file. While previously these were different scenarios that used different scripts, the refactored code also introduced all these modes of operation for both the key and the plaintext values under the same framework. In addition, some attacks may only require some points of interest, these can be defined in the new script so the trace output consists only of these points.

Finally, the previous code saved the produced files in existing folders that had to be manually created. Running the same script twice would inevitably overwrite existing data if the script was not moved into a new folder. In order to ensure all data was preserved, a feature was newly introduced: every time the script is run, it creates the entire necessary folder structure to organize and preserve the captured data. Furthermore, and perhaps most importantly, the folder names are timestamped and, by default, contain a log file of all the settings used and of any warning set off by the script during capture. This ensures all data captured is completely characterized, providing easy traceability and result transparency.

In conclusion, this work has introduced improvements in the capture script that allow for effortless consistent captures under controlled conditions, now available at the research facility for further work.

Attack modules

There are two main attack modules: the correlation attack module, first implemented by João Lopes [36] and the template attack module, which was newly developed. These modules are implemented as sets of MATLAB scripts, chosen for the ease of manipulation of huge matrix datasets necessary to perform the attacks.

The correlation attack module allows for a user to specify the number of traces to read. First, for that number of traces, the module reads the relevant information — for a first round attack, this would be the plaintexts and the respective power traces. After loading the necessary information, the attack begins with a loop that iterates every partial key, trying to break each one at a time. Then, for each subkey hypothesis, the power model is generated for all traces and compared with the actual power consumption using an implementation of the correlation function optimized for matrices by João Lopes. These operations were originally implemented using loops. However, in this work, a new vectorized code

architecture was implemented that better utilizes the matrix processing power of MATLAB, eliminating some of the iterative loops that tested each trace individually, instead calculating the results for a matrix of all traces simultaneously.

A bug was discovered in the original code for the first round correlation attack that slightly biased the results obtained. Due to the large amounts of data used and to save memory usage, the original version by João Lopes [36] implemented the key addition result as an 8-bit positive integer vector, which in principle is sufficient to cover every single value of that byte, from 0-255. However, due to the fact that MATLAB indexation of matrices starts at one, when applying the SBOX transformation, a shift of one is applied to these values by adding 1. However, this means the integer overflows, since it cannot represent 256, in fact, the resulting mapping will only cover 254 values, from 1-255. To solve this problem, the value is now cast into the default MATLAB type (double precision) before the operation is executed, to ensure no overflows are possible for attacks targeting one byte. This correction was also incorporated in Lopes' work.

An implementation was developed which runs this attack several times, picking a certain number of random traces from the larger captured set. This simulates the execution of several attacks of the same number of traces, under the same conditions, and allows the attacker to establish measures of confidence and success on the results.

The template attack module was newly implemented from scratch using the concepts presented in chapter 3. This module is composed of two major parts, a set of scripts responsible for training template models, and another set responsible for attacking traces using an existing model.

For training templates, one needs to capture a certain number of samples for each possible intermediate value, preferably without being biased to any key or plaintext in particular. In the template attack module, an attacker may start by using the script that generates random plaintext-key pairs with a certain number of samples for each intermediate value in each partial key in a non-sequential manner. The two files generated, for keys and plaintexts, are then used in the capture process with the Trace Capture module.

To build the template model from the traces, a script was developed which takes the points of interest of the power traces and their corresponding key and plaintext. In order to simplify this first implementation, only univariate models are built. The arithmetic mean and standard deviation are calculated for each set of identical intermediate values, as estimators of the mean and variance of their power distribution. The model is saved in the disk as a .MAT file containing a matrix of mean and variance pairs for every partial key.

Finally, an attack can be carried out. Using a distinct set of traces taken from a device hiding a fixed key, a script was developed which uses the previous template model and, for every partial key, loops over key hypotheses and calculates the likelihood of the observed interest points in the power traces with respect to the normal distribution of the templates. Finally, a maximum likelihood distinguisher is applied by calculating, for each key hypothesis, the product of all observed likelihoods. The script outputs the best matches for each subkey.

Embedded software

There are two main devices used in the experimental work: the FPGA and the smartcard. In the case of the FPGA, there is an existing AES implementation of a cipher provided by the SAKURA team [37] which includes a trigger signal suitable to synchronize the signal acquisition of the oscilloscope. The software is composed of two parts, one for the control chip and the other, which incorporates the ciphering algorithm, aimed at the main chip. These are loaded from the computer into the FPGA using the JTAG interface in each of the FPGA chips. This implementation is well-suited for this work and is the one used throughout the FPGA experiments.

Regarding the smartcard, the software used was developed by João Lopes [36] and is based on the tiny-AES implementation by GitHub username *kokke*. Building on top of the cipher code, João developed a trigger signal, which allows for time-aligned trace capture. The implementation of the trigger is simple: as the operation of interest in AES is executed, one of the auxiliary pins of the smartcard is set to HIGH for its duration.

For communication, João Lopes implemented the ISO/IEC 7816 standard, defining several commands which can be sent by the computer. Each command is sent using a APDU (*application protocol data unit*), which is composed by an instruction code, defining the operation to be executed, and the amount of data that will be sent. There are instruction codes which allow for all of the most important commands required to run the cipher in the smartcard, namely, to send a plaintext to the smartcard, start the cipher, retrieve the ciphertext and set a new key. Having defined these APDUs, it is possible to use a serial communication interface to send the commands using the Python script, as previously described.

5.2 FPGA Results

In this section, the results gathered from the FPGA are detailed. Successful attacks were possible, improving on the existing results of João Lopes [36] due to new features in the trace capture process. First, the baseline results are detailed and the possible experimental setups are compared. Settling on an experimental setup, several signal processing techniques are applied in order to investigate the possibility of further improving results.

Using the amplified port from the FPGA, it is possible to capture traces with no additional equipment. This will be the first setup studied. Figure 5.6 illustrates a typical trace captured from the FPGA, in this case, using the aforementioned internal amplifier, during 8 microseconds. It is possible to detect a repeating pattern of sudden voltage drops, which correspond to sudden increases in consumption of power by the FPGA. The FPGA computes each round of the cipher in this short amount of time, thus, the 10 rounds of AES correspond to these 10 signal fluctuations. The attacks will target the first round of AES, which corresponds to the first element of the repeating pattern. In particular, the values targeted by the differential attack are the intermediate results of the SubBytes transformation, preceded by the key addition.

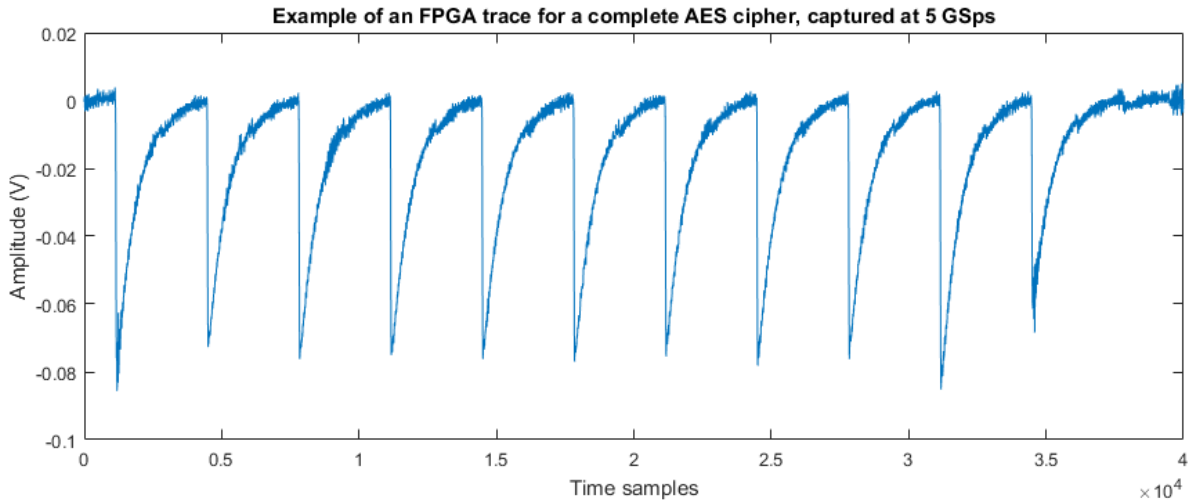

Figure 5.6: FPGA trace for a complete AES run, captured during 8 μs with a 3% pretrigger and sampling frequency of 5GHz, using the internal amplifier.

5.2.1 Correlation attacks: setup comparison

The capture of 200 000 traces, with sampling frequency 2.5 GHz, was conducted in order to proceed with attacking the device. The results of correlation attacks using this setup, using the Hamming Weights model, are laid out in Table 5.1, using the success metrics previously defined.

Table 5.1: Results of correlation attacks to the first round of AES in the FPGA, using the internal amplifier, with the Hamming weights model (averages of 400 retrials, sampled from 200 000 traces)

Number of traces iAMP	SR 16 subkey average	SR^{CK} complete key	Correct normalized dist. 16 subkey average	Correct subkey position 16 subkey average
1000	0.005 ± 0.001	0.000 ± 0.000	-0.243 ± 0.001	121.716 ± 0.919
13000	0.135 ± 0.004	0.000 ± 0.000	-0.132 ± 0.002	57.247 ± 0.814
26000	0.374 ± 0.006	0.000 ± 0.000	-0.017 ± 0.002	24.065 ± 0.553
38000	0.566 ± 0.006	0.000 ± 0.000	0.068 ± 0.002	12.114 ± 0.384
50000	0.703 ± 0.006	0.000 ± 0.000	0.138 ± 0.002	5.739 ± 0.231
56000	0.769 ± 0.005	0.010 ± 0.005	0.170 ± 0.003	5.085 ± 0.239
60000	0.800 ± 0.005	0.010 ± 0.005	0.189 ± 0.003	3.832 ± 0.188
65000	0.831 ± 0.005	0.015 ± 0.006	0.214 ± 0.003	3.099 ± 0.149
70000	0.849 ± 0.004	0.030 ± 0.009	0.234 ± 0.003	2.645 ± 0.119
75000	0.876 ± 0.004	0.077 ± 0.013	0.256 ± 0.003	2.230 ± 0.113
80000	0.891 ± 0.004	0.095 ± 0.015	0.273 ± 0.003	1.987 ± 0.093
86000	0.912 ± 0.004	0.135 ± 0.017	0.292 ± 0.003	1.717 ± 0.071
92000	0.922 ± 0.003	0.163 ± 0.018	0.312 ± 0.003	1.511 ± 0.055
99000	0.938 ± 0.003	0.237 ± 0.021	0.332 ± 0.003	1.385 ± 0.039
106000	0.942 ± 0.003	0.253 ± 0.022	0.350 ± 0.003	1.306 ± 0.034
114000	0.950 ± 0.003	0.302 ± 0.023	0.374 ± 0.003	1.200 ± 0.018
123000	0.959 ± 0.002	0.400 ± 0.025	0.394 ± 0.003	1.149 ± 0.015
132000	0.961 ± 0.002	0.403 ± 0.025	0.414 ± 0.003	1.123 ± 0.012
141000	0.965 ± 0.002	0.450 ± 0.025	0.428 ± 0.003	1.100 ± 0.010

These results fail to achieve perfect success for the complete key, with a maximum success rate for the complete key of 45%. However, all the metrics follow the expected trend over the increasing number of traces, with increasing success rates (both for the subkeys and complete attack), increasing normalized distances and decreasing subkey position. This is the expected result for a working attack, which

would approximately reach a success rate of unity if the number of traces used had been sufficiently increased.

It is also possible to use the external amplifier available at the laboratory instead of the internal amplifier. In that case, the DC block must also be used. The external amplifier is able to outperform the amplification of the internal one, and is adjustable so that the signal occupies exactly the range of the oscilloscope, taking advantage of the entire 8 bits of sample resolution in that range. Accordingly, the amplifier was set at 12.04V so as to occupy the entire range of the oscilloscope. According to the data sheet, the gain of the amplifier should be typically 20 dB when powered with 12V. After mounting this setup and capturing 200 000 traces, the attacks were mounted with identical conditions to the previous attacks, but due to time constraints, reducing the number of retrials to 50. The results of correlation attacks using a Hamming weights model against this setup are shown in Table 5.2.

Table 5.2: Results of correlation attacks to the first round of AES in the FPGA, using the external amplifier, with the Hamming weights model (averages of 50 retrials sampled from 200 000 traces)

Number of traces AMP	SR 16 subkey average	SR^{CK} complete key	Correct normalized dist. 16 subkey average	Correct position (improvement vs. iAMP)
1000	0.00 ± 0.00	0.00 ± 0.00	-0.26 ± 0.00	121.15 ± 2.58 (0.5%)
13000	0.14 ± 0.01	0.00 ± 0.00	-0.14 ± 0.01	54.00 ± 2.22 (5.7%)
26000	0.42 ± 0.02	0.00 ± 0.00	-0.01 ± 0.01	19.90 ± 1.40 (17.3%)
38000	0.61 ± 0.02	0.00 ± 0.00	0.08 ± 0.01	10.32 ± 1.00 (14.8%)
50000	0.75 ± 0.02	0.00 ± 0.00	0.15 ± 0.01	4.98 ± 0.60 (13.2%)
56000	0.79 ± 0.01	0.00 ± 0.00	0.19 ± 0.01	4.29 ± 0.62 (15.6%)
60000	0.81 ± 0.01	0.02 ± 0.02	0.20 ± 0.01	2.61 ± 0.26 (31.8%)
65000	0.83 ± 0.01	0.00 ± 0.00	0.22 ± 0.01	2.79 ± 0.28 (10.1%)
70000	0.86 ± 0.01	0.04 ± 0.03	0.25 ± 0.01	2.06 ± 0.20 (21.9%)
75000	0.89 ± 0.01	0.06 ± 0.03	0.28 ± 0.01	2.36 ± 0.39 (-5.9%)
80000	0.89 ± 0.01	0.08 ± 0.04	0.29 ± 0.01	2.00 ± 0.25 (-0.6%)
86000	0.91 ± 0.01	0.10 ± 0.04	0.31 ± 0.01	1.76 ± 0.19 (-2.5%)
92000	0.92 ± 0.01	0.18 ± 0.05	0.34 ± 0.01	1.47 ± 0.12 (3.0%)
99000	0.92 ± 0.01	0.16 ± 0.05	0.35 ± 0.01	1.34 ± 0.08 (3.1%)
106000	0.95 ± 0.01	0.34 ± 0.07	0.38 ± 0.01	1.21 ± 0.05 (7.3%)
114000	0.95 ± 0.01	0.30 ± 0.07	0.39 ± 0.01	1.18 ± 0.05 (1.7%)
123000	0.96 ± 0.01	0.44 ± 0.07	0.42 ± 0.01	1.08 ± 0.02 (6.3%)
132000	0.96 ± 0.01	0.48 ± 0.07	0.44 ± 0.01	1.12 ± 0.03 (0.3%)
141000	0.97 ± 0.01	0.48 ± 0.07	0.46 ± 0.01	1.07 ± 0.02 (3.0%)

Again, these results show a good progression of the metrics. The comparison of the different setups will follow, but first, other possible setup involves combining both amplifiers, the internal amplification and further external amplification, to produce a twice-amplified trace. As before, the amplifier was set at 12.5 V, a small increase in gain so as to occupy the entire range of the oscilloscope. A total of 200 000 traces were collected and Hamming weight model correlation attacks were run. Their results are displayed in Table 5.3.

These three correlation attacks for the different setups were compared, as presented in Figures 5.7 and 5.8, for the average correct key position and correct key normalized distance.

When comparing the internal and external amplifiers (iAMP vs AMP+DCB), the additional amplification provided by the external amplifier moderately improved the results of the attacks. This setup displays the lowest average correct subkey position and the highest normalized distance. This implies

Table 5.3: Results of correlation attacks to the first round of AES of the FPGA, using the internal amplifier and external amplifier in series, with the Hamming weights model (averages of 50 retrials, sampled from 200 000 traces)

Number of traces	SR 16 subkey average	SR^{CK} complete key	Correct normalized dist. 16 subkey average	Correct subkey position (improvement vs. iAMP)
1000	0.00 ± 0.00	0.00 ± 0.00	-0.24 ± 0.00	120.60 ± 2.56 (0.91%)
13000	0.11 ± 0.01	0.00 ± 0.00	-0.13 ± 0.00	56.25 ± 2.30 (1.75%)
26000	0.36 ± 0.02	0.00 ± 0.00	-0.02 ± 0.01	20.85 ± 1.39 (13.38%)
38000	0.56 ± 0.02	0.00 ± 0.00	0.05 ± 0.01	11.17 ± 0.99 (7.77%)
50000	0.70 ± 0.02	0.00 ± 0.00	0.12 ± 0.01	5.77 ± 0.66 (-0.57%)
56000	0.75 ± 0.02	0.00 ± 0.00	0.15 ± 0.01	4.12 ± 0.44 (19.00%)
60000	0.78 ± 0.01	0.00 ± 0.00	0.15 ± 0.01	5.13 ± 0.64 (-33.88%)
65000	0.81 ± 0.01	0.00 ± 0.00	0.19 ± 0.01	3.64 ± 0.51 (-17.45%)
70000	0.83 ± 0.01	0.02 ± 0.02	0.21 ± 0.01	2.79 ± 0.35 (-5.56%)
75000	0.86 ± 0.01	0.02 ± 0.02	0.22 ± 0.01	2.61 ± 0.33 (-16.88%)
80000	0.88 ± 0.01	0.02 ± 0.02	0.24 ± 0.01	2.74 ± 0.32 (-37.83%)
86000	0.88 ± 0.01	0.00 ± 0.00	0.27 ± 0.01	1.98 ± 0.26 (-15.40%)
92000	0.88 ± 0.01	0.04 ± 0.03	0.29 ± 0.01	1.72 ± 0.16 (-14.05%)
99000	0.92 ± 0.01	0.12 ± 0.05	0.31 ± 0.01	1.66 ± 0.14 (-20.05%)
106000	0.93 ± 0.01	0.22 ± 0.06	0.32 ± 0.01	1.43 ± 0.15 (-9.21%)
114000	0.94 ± 0.01	0.24 ± 0.06	0.35 ± 0.01	1.23 ± 0.05 (-2.72%)
123000	0.94 ± 0.01	0.14 ± 0.05	0.36 ± 0.01	1.31 ± 0.07 (-14.22%)
132000	0.95 ± 0.01	0.22 ± 0.06	0.39 ± 0.01	1.20 ± 0.04 (-6.97%)
141000	0.95 ± 0.01	0.32 ± 0.07	0.41 ± 0.01	1.22 ± 0.10 (-11.32%)

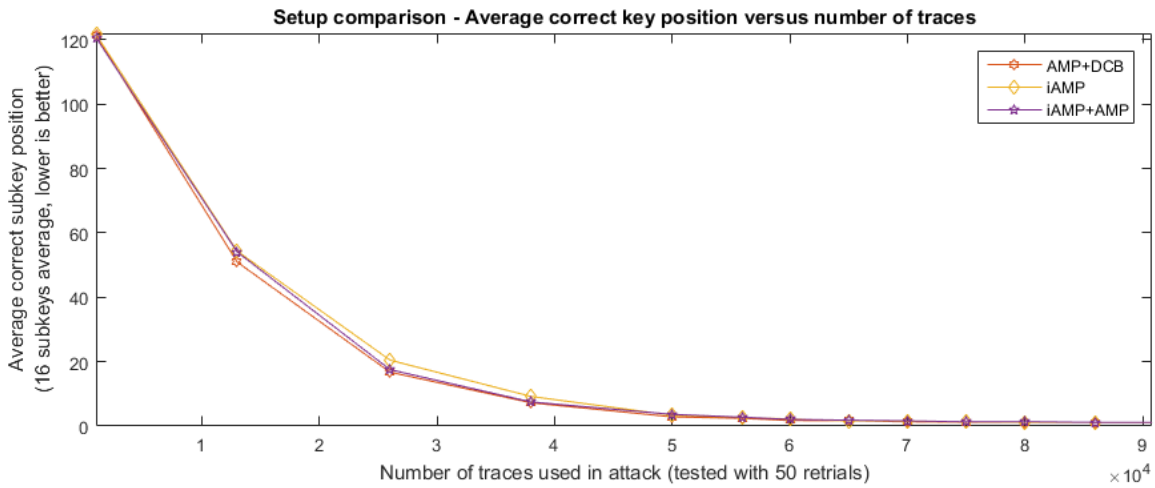

Figure 5.7: Results of average correct partial key position for correlation attacks with different number of traces, for 50 retrials with the Hamming weights model.

that attacks run using the traces from the external amplifier place the correct candidate more frequently in the top position, and that the competing hypotheses are consistently more distant from the correct hypotheses, when comparing the setups.

For the combination of both amplifiers (iAMP+AMP), the results were not significantly better than using only the external amplifier. In fact, it only performed equally well for small numbers of traces, being outpaced by both the internal amplifier (denoted iAMP) and the external amplifier for the range of number of traces that leads to reliable attack results (above 50 000 traces).

The results of both benchmarks indicate the best setup is the one using the external amplifier (combined with a mandatory DC block, denoted as AMP+DCB).

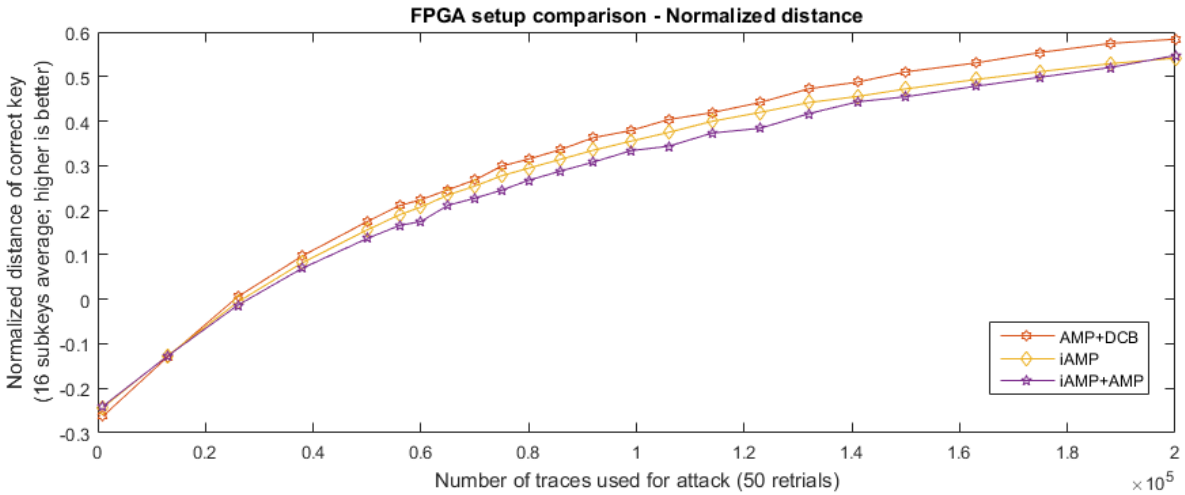

Figure 5.8: Results of correct partial key normalized distance for correlation attacks with different number of traces, for 50 retrials with the Hamming weights model.

To conclude, the flexibility of the external amplifier allows the attacker to adjust the signal of traces so as to occupy the maximum range of the oscilloscope, and the additional gain provides a small improvement. For these reasons, the FPGA attacks will proceed using this setup. These results are consistent with João Lopes' [36] conclusions for these attacks.

5.2.2 Correlation attacks: trace filtering

Taking into account the setup results and the side-channel theory, two signal processing techniques based on filtering were considered in order to further improve the results of the best setup, especially knowing most of the leakage energy is located in the lower frequencies.

Digital filters are important tools that process a signal in some way, modifying certain properties of a signal, such as its frequency and/or phase content. To remove the higher frequencies, it is necessary to implement a filter that only lets the lower frequencies pass through, thus its name, *low-pass filter*.

Besides removing interfering noise in frequencies not relevant to the attack, the digital filters also decrease the error of quantization, that is to say, the rounding error in each sample. This is due to the fact that, when applying a digital filter, each sample is computed using the neighbouring samples, which increases the precision of every sample.

The filters that were designed are called Finite Impulse Response (FIR) filters. An FIR filter computes each output sample as a normalized sum of a number of preceding sample inputs, called *taps*. The filters are calculated starting with the first time sample, and iterate over the entire trace, always calculating the next time sample using the previous N samples, where N is the number of taps.

One of the simplest filters that can be designed is called a moving average filter, which calculates each new sample as the average of a certain quantity of preceding samples, reducing both the rounding error of each sample and the high frequency variations. However, more advanced filters can be designed with better frequency response. In particular, a window function can be used, which tapers the sample inputs of the filters to avoid sudden changes in the output, leading to a better frequency response. In

that way, if a new sample is added to the end of the normalized sum as the farthest preceding sample, the window function ensures that its contribution is only significant as it travels the filter memory.

Several low-pass digital filters were designed. Digital filters operate on normalized frequencies, which depend on the sampling frequency of the original signal. A normalized frequency of 1 corresponds to half the sampling frequency, which is also the limit for the frequency content of the signal sampled without distortion. Using MATLAB's fir1() function, the coefficients of several filters were calculated with a Hamming window and 2400 taps. The first low-pass filter has a cutoff normalized frequency of 0.001, and the remaining filters have cutoff spaced 0.001 apart. In total, 50 filters were designed in this way, covering the spectrum from 0.001 to 0.050 of the total possible frequency content of the digital signal. The frequency response for two examples of the filters designed are shown in Figure 5.9.

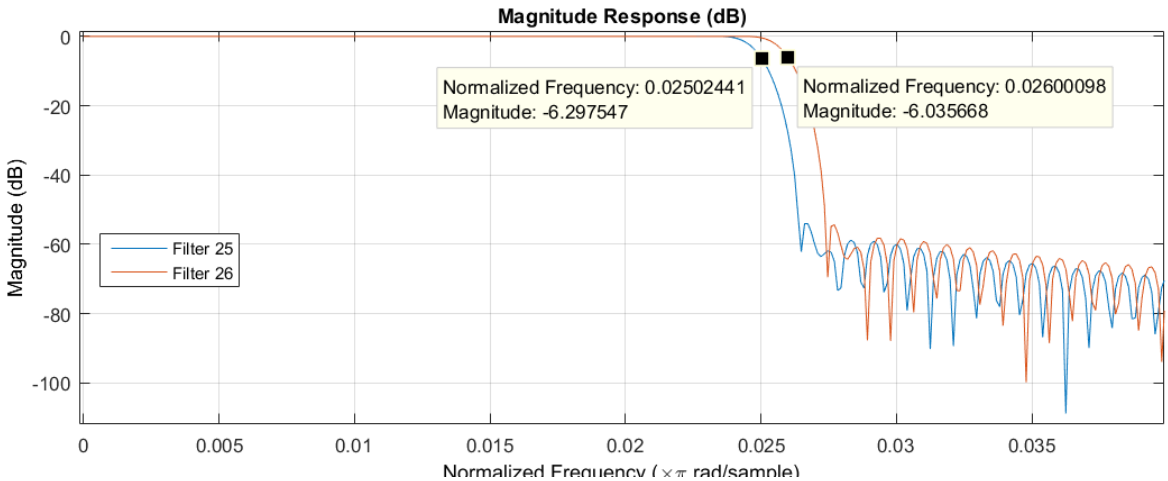

Figure 5.9: Example of two consecutive filters generated. The filters generated have cutoffs shifted 0.001 in the normalized frequency, at -6 dB.

Trace filtering is the technique characterized by the application of a low-pass filter to the traces in order to filter sudden signal variations mostly caused by noise. An attacker may have a clone of the device she or he intends to attack, and may use it to identify which low-pass filter provides optimal results. Figure 5.10 depicts the result of processing traces using a low-pass filter. In this case, the different low-pass filters were tested using the 50 available low-pass filters in order to evaluate which filter provided the best results.

A initial frequency sweep was carried out in traces captured with a sampling frequency of 5GHz. The same correlation attack with a Hamming weights model, under the exact same conditions, was carried out after processing the traces with each one of the available filters. Using the aforementioned filters, it is possible to test the effect of low-pass filters with cutoff frequencies starting at $5000 \div 2 \times 0.001 = 2.5$ MHz, up to $5000 \div 2 \times 0.001 \times 50 = 125$ MHz. The results of the frequency test are shown in Figure 5.11. The sweep suggests the optimal cutoff frequency to be between 10 and 20 MHz. Increasing the high-frequency content of the filtered signal is possible by allowing a higher cutoff frequency, but this seems to negatively impact the outcome of the attack as shown in the 70 to 100 MHz area, where the distance of the correct key differs from the distance of the best guess, implying the attack is making fewer correct guesses.

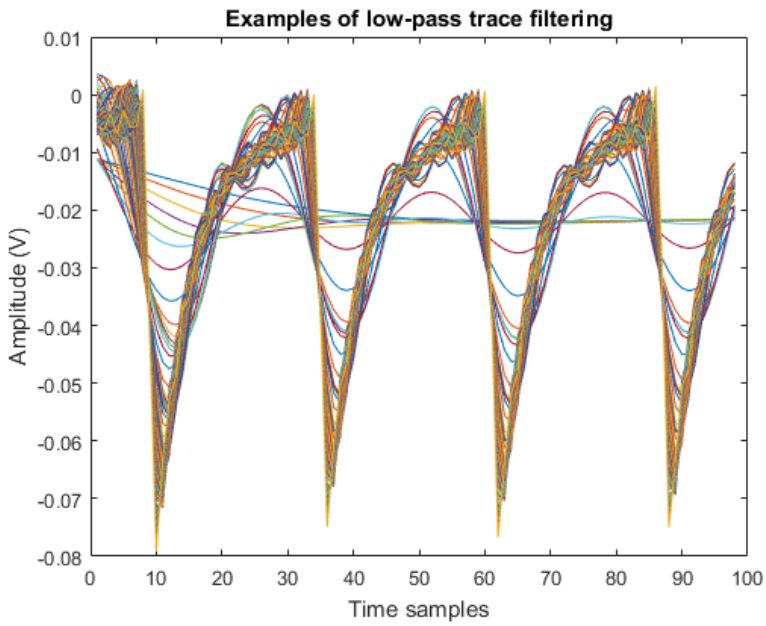

Figure 5.10: Example of the same trace, processed by several different low-pass filters with decreasing cutoff frequencies.

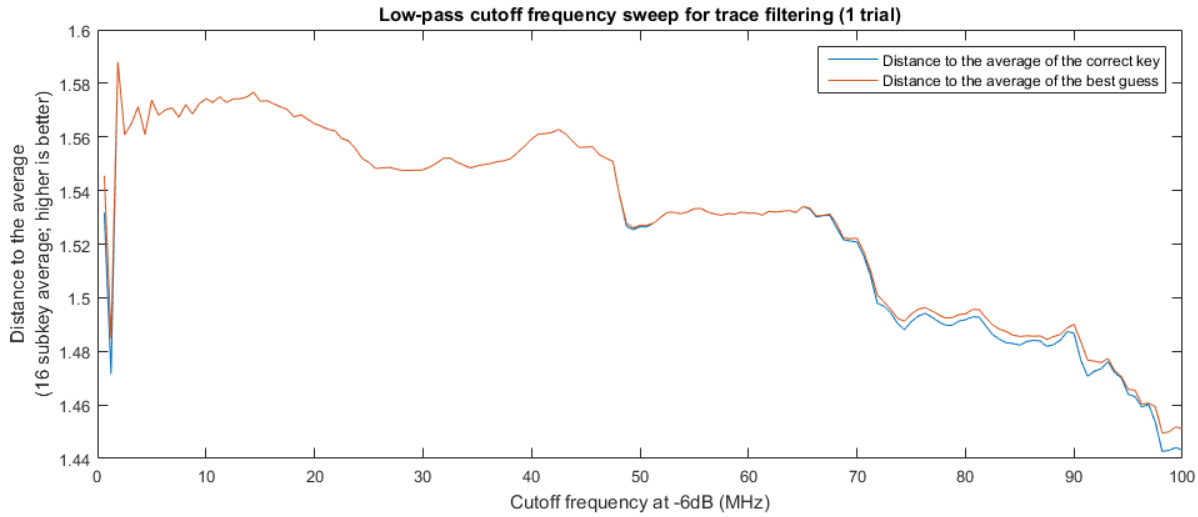

Figure 5.11: Results of correlation attacks run against traces processed using filters of different cutoff frequencies (1 trial with the Hamming weights model, 2.5 to 100 MHz shown).

Focusing on the area of interest, a more careful analysis was conducted, depicted in Figure 5.12. Instead of running one trial, 50 trials were conducted for each of the 50 filters. This time, the sampling frequency of the captured traces was halved, which corresponds to a halving of each of the cutoff frequencies with respect to the original signal. In that case, the correlation attacks were studied with cutoff frequencies between 1.25 MHz and 62.5 MHz.

The best results were obtained at 3.75 MHz for the 35 000 traces and 13.75 MHz for the 50 000 traces. The results for these filtered traces are displayed in Table 5.4, and show modest improvements in the success rate and the correct key position.

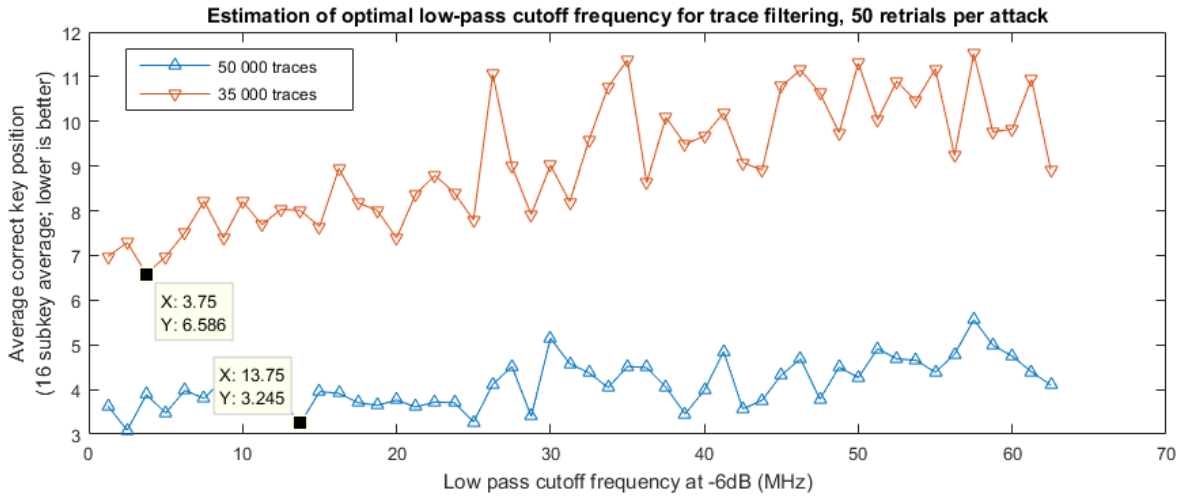

Figure 5.12: Results of correlation attacks run against traces processed using filters of different cutoff frequencies (50 trials with the Hamming weights model, 1.25 to 62.5MHz).

Table 5.4: Results of trace filtering in correlation attacks to the first round of AES with the Hamming weights model, for 50 retrials (sampled from 200 000 traces), using the optimal filters, compared to the original results.

Experimental setting	SR 16 subkey average	SR^{CK} complete key	Normalized distance 16 subkey average	Correct subkey position (improvement vs. regular)
50 000 traces regular external amplifier	0.75 ± 0.02	0.00 ± 0.00	0.15 ± 0.01	4.98 ± 0.60
50 000 traces low pass filtered cutoff -6 dB: 3.75 MHz	0.75 ± 0.02	0.00 ± 0.00	0.23 ± 0.01	3.89 ± 0.43 (21.9%)
50 000 traces low pass filtered cutoff -6 dB: 13.75 MHz	0.77 ± 0.01	0.00 ± 0.00	0.23 ± 0.01	3.24 ± 0.32 (34.9%)

In conclusion, while the improvement is not substantial, it suggests that the majority of the leakage is located in the lowest frequencies. In the next section, another approach is trialled to extract the most of the lower frequency leakage.

5.2.3 Correlation attacks: correlation filtering

Correlation filtering is a proposed technique that takes the correlation over time of every value hypothesis and filters sudden variations, to remove sporadic correlation peaks from competing hypotheses and to ensure that the correlation smoothly peaks at the time instant when the operation is performed. The original correlation trace, as shown in Figure 5.13, can have a considerable amount of noise. For the typical criteria of choosing the correlation maxima, the noise fluctuations may easily be preferred instead of a correct key. Depending on the device under attack, the attacker might want to choose a low-pass filter that preserves the leakage while improving the results.

The application of a correlation filter was originally inspired in the selection of points of interest for template attacks. It is commonly known that nearby points of interest are redundant, that is to say, if one operation is well represented by two time samples that are very close, they are likely extracting the same

leakage — this is very clear in Figure 5.13 where the correlation peak is not only located in a single time sample, but in a small hill.

The correlation filter not only de-emphasizes the sudden variations caused by noise, but it also mutually emphasizes the redundant points of interest. Due to the construction of a low-pass filter, the value of each time sample is affected by the neighbouring values in order to reduce sudden fluctuations, which means that neighboring leakage points will be mutually reinforced.

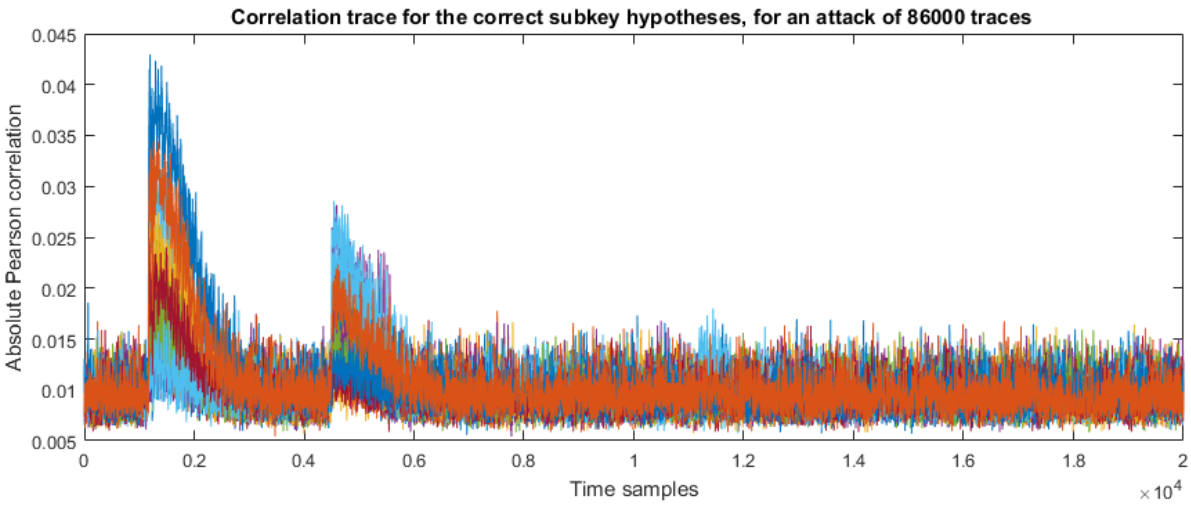

Figure 5.13: Raw correlation trace for the 16 correct values of the subkeys in a 86 000-trace FPGA correlation attack.

In Figure 5.14, the correlation was low-pass filtered after computation, and the maxima of filtered traces no longer includes any significant noise. In that figure, it is also possible to get a clear picture of the points of interest in this attack. Since the attack is targeted at the first round, the first round is the main source of leakage. The second round also leaks information for the attack to the first round, possibly due to the transitions of the registers between the first and second round.

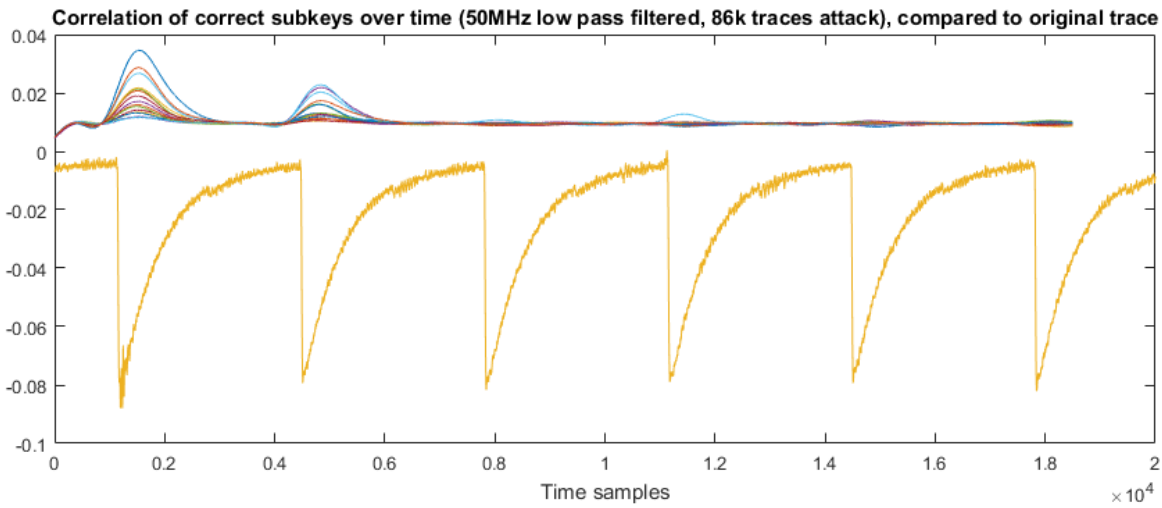

Figure 5.14: Filtered correlation traces of the correct hypotheses (top) compared to the power trace in a 86 000-trace FPGA correlation attack (bottom).

As in the trace filtering case, a study of the optimal filter to use was conducted by testing the effect of

filtering the traces with each of the available filters using correlation attacks under the same conditions. Figure 5.15 displays the results of the cutoff frequency sweep. Immediately, the average position for the correct key is noticeably lower, either using 35 000 or 50 000 traces. Whereas in the regular attack, 50 000 traces displayed an average correct position of approximately 5, the attacks with correlation filtering fluctuate between 1.5 and 2, corresponding to more than a 2-fold improvement.

Two possible values were selected for the cutoff frequency taking into account the local minima in the correct key position: 18.75 MHz and 13.75 MHz (the latter, in fact, the same frequency that was selected for trace filtering).

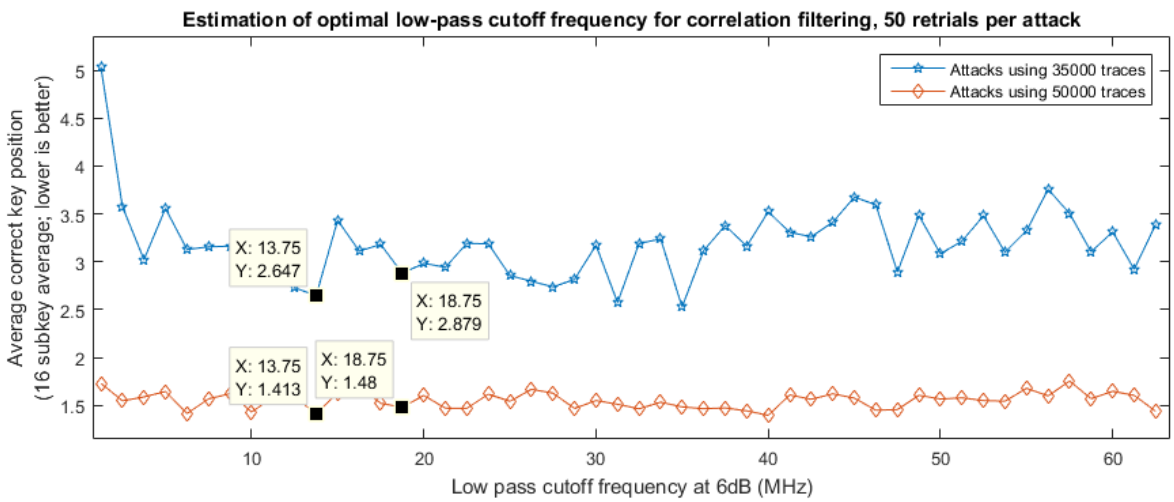

Figure 5.15: Results of correlation attacks with the correlation trace processed using filters of different cutoff frequencies (50 trials, with the Hamming weights model, 1.25 to 62.5MHz) in the externally amplified setup.

For such noticeable improvements, an experiment was conducted where the attacks were run for several number of traces while applying one of the two correlation filters. This new run uses traces randomly selected from the original pool and exhibits a small deviation from the results of the cutoff characterization. The results of the correlation filtering using the 18.75 MHz filter are presented in Table 5.5 and can be compared to the original setup and the alternative filter in Figure 5.16. Both filters display a similar considerable performance improvement when compared to the original trace.

Table 5.5: Results of correlation attacks with the Hamming weights model, to the first round of AES in the FPGA using the external amplifier and applying correlation filtering with cutoff frequency at -6 dB of 18.75 MHz (averages of 50 retrials sampled from 200 000 traces)

Number of traces	SR 16 subkey average	SR^{CK} complete key	Normalized distance 16 subkey average	Correct subkey position (improvement vs. external AMP)
1000	0.01 ± 0.00	0.00 ± 0.00	-0.50 ± 0.01	112.24 ± 2.61 (7.35%)
13000	0.28 ± 0.02	0.00 ± 0.00	-0.13 ± 0.01	25.82 ± 1.52 (52.19%)
26000	0.52 ± 0.02	0.00 ± 0.00	0.06 ± 0.01	8.60 ± 0.71 (56.79%)
38000	0.70 ± 0.02	0.00 ± 0.00	0.19 ± 0.01	3.90 ± 0.42 (62.25%)
50000	0.81 ± 0.01	0.00 ± 0.00	0.27 ± 0.01	2.38 ± 0.27 (52.19%)
56000	0.86 ± 0.01	0.06 ± 0.03	0.32 ± 0.01	2.26 ± 0.34 (47.41%)
60000	0.89 ± 0.01	0.02 ± 0.02	0.35 ± 0.01	1.84 ± 0.16 (29.51%)
65000	0.89 ± 0.01	0.08 ± 0.04	0.37 ± 0.01	1.62 ± 0.13 (41.77%)
70000	0.90 ± 0.01	0.10 ± 0.04	0.39 ± 0.01	1.42 ± 0.08 (31.30%)
75000	0.91 ± 0.01	0.10 ± 0.04	0.41 ± 0.01	1.29 ± 0.05 (45.21%)
80000	0.93 ± 0.01	0.16 ± 0.05	0.44 ± 0.01	1.51 ± 0.11 (24.70%)
86000	0.94 ± 0.01	0.30 ± 0.07	0.47 ± 0.01	1.31 ± 0.07 (25.78%)
92000	0.94 ± 0.01	0.22 ± 0.06	0.48 ± 0.01	1.22 ± 0.05 (17.05%)
99000	0.95 ± 0.01	0.34 ± 0.07	0.51 ± 0.01	1.20 ± 0.04 (10.61%)
106000	0.95 ± 0.01	0.34 ± 0.07	0.52 ± 0.01	1.16 ± 0.04 (4.33%)
114000	0.96 ± 0.01	0.34 ± 0.07	0.56 ± 0.01	1.13 ± 0.04 (4.24%)
123000	0.96 ± 0.01	0.34 ± 0.07	0.57 ± 0.01	1.12 ± 0.03 (-3.95%)
132000	0.97 ± 0.01	0.48 ± 0.07	0.59 ± 0.01	1.07 ± 0.02 (4.58%)
141000	0.96 ± 0.01	0.42 ± 0.07	0.61 ± 0.01	1.07 ± 0.02 (-0.59%)

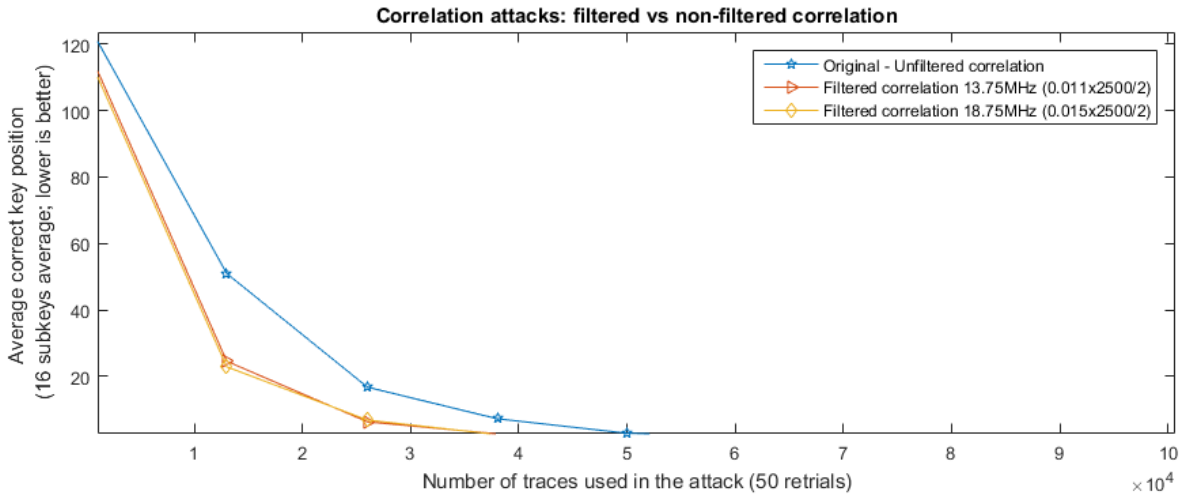

Figure 5.16: Results of average correct partial key position for correlation attacks with different number of traces, with and without correlation filtering, for 50 retrials using the Hamming weights model in the externally amplified setup.

Unfortunately, it was not possible to run well-designed template attacks against the FPGA setup. Due to a particularity of the FPGA programming, each time a new key is set, the first trace observed has significantly less leakage than the next. Template attacks rely on a characterization of the device which is done using randomly-selected keys for every trace. Given the visible changes on the traces when the keys were changed, there was no adequate guarantee of consistency in the templates and the template attack efforts were discontinued for this device.

In summary, the correlation filtering successfully improved the results of the correlation attack. It offered a statistically significant improvement over the unprocessed attacks and the attacks with filtered

traces. It is interesting to note that the filters selected for both were similar, which suggests that the filtration of correlation and traces are not independent. In a small scale test, the usage of both techniques simultaneously — correlation and trace filtering — did not improve the results, which seems to support this hypothesis.

5.3 Smartcard Results

The smartcard attack results are presented in this section, starting with an experiment using the correlation attack already explored with the FPGA, in addition to an experiment with template attacks. Successful attacks were achieved with a surprisingly low number of traces, requiring a typical attacker seconds to capture the signals, and a few minutes to process the data and retrieve the correct key.

For the smartcard setup, the extension board SAKURA-W does not include an internal amplifier. The advantages of using amplification are clear from the previous tests using the FPGA, so experiments are conducted using the external amplifier powered at 15 V, for a minimum amplification of 20 dB and a typical amplification of 23 dB, according to the manufacturer's data sheet.

A typical smartcard trace is displayed in Figure 5.17. Right away, one may notice the significant increase in the trace power. The FPGA trace amplitude is on the order of tens of millivolts, while the smartcard trace is on the order of volts. While this may facilitate an attack, higher power signals do not necessarily imply more leakage. In fact, Figure 5.17 only shows the detail of a few smartcard instructions which result in the periodic fluctuation depicted. The actual entire trace only captures the first round, and is 875 000 samples long, composed of a indiscernible sequence of these operations for a duration of 350 microseconds. As such, the traces display better detail, both in time and amplitude resolution.

In total, 1 000 traces were captured for each different experiment. As before, all attacks are targeted at the first round of AES, in particular, the intermediate result after the SubBytes transformation and key addition.

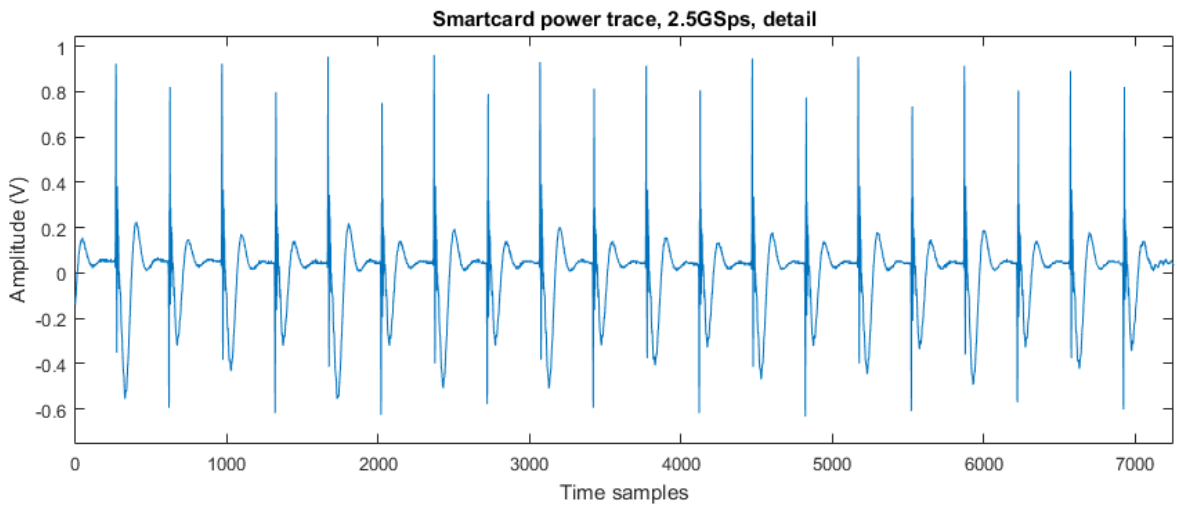

Figure 5.17: Detail of a smartcard trace highlighting the individual operations, captured with a 3% pre-trigger and with sampling frequency of 2.5GHz, using the external amplifier.

5.3.1 Correlation attacks

The correlation attacks using the Hamming weights model were conducted on the 1 000 captured traces. The results of correlation attacks are displayed in Table 5.6. There is, in fact, a considerable reduction of the number of traces needed to attack the device when compared to the FPGA. Whereas in the FPGA the success rate of the complete key, for the number of traces presented, never reached 100%, that is achieved in the smartcard attacks after using a mere 50 traces. Recall however that these traces have much more data, since the calculation of the round takes longer and more time samples need to be captured.

Table 5.6: Results of power correlation attacks to the first round of AES in the smartcard using the external amplifier, using the Hamming weights model (averages of 50 retrials sampled from 1 000 traces)

Number of traces	SR 16 subkey average	SR^{CK} complete key	Correct normalized distance 16 subkey average	Correct subkey position 16 subkey average
10	0.00 ± 0.00	0.00 ± 0.00	-0.03 ± 0.00	81.16 ± 2.22
20	0.54 ± 0.02	0.00 ± 0.00	0.00 ± 0.00	12.02 ± 1.05
30	0.95 ± 0.01	0.48 ± 0.07	0.08 ± 0.00	1.24 ± 0.07
40	1.00 ± 0.00	0.98 ± 0.02	0.17 ± 0.00	1.00 ± 0.00
50	1.00 ± 0.00	1.00 ± 0.00	0.25 ± 0.00	1.00 ± 0.00

5.3.2 Template attacks

In this section, the results of the template attacks to the smartcard are presented. In this case, univariate template attacks were conducted using the developed software.

In order to carry out the experiment, the templates have to be generated. For that, the points of interested were selected based on the maximum correlation observed in the previous correlation attacks. For an attacker that does not have the relevant points of interest, she or he may instead run a t-test in order to evaluate the leakage points.

A total of 1 000 traces were captured employing random keys, so as to simulate an attacker generating templates to exploit a device that can hide any possible key. Taking the values of the plaintext bytes and the partial keys, the template parameters for each of the intermediate calculation bytes were calculated — average and standard deviation. This results in a $2 \times 16 \times 256$ matrix containing the pair of parameters for each of the 256 possible values, for every byte of intermediate calculation, out of 16. Using the same reasoning, one can expect each of the intermediate values to take into account, on average, $1\ 000 \div 256 = \ 3.9$ experimental samples. Figure 5.18 displays the templates for one of the bytes of the intermediate state being attacked. Notice how the templates generally follow the Hamming Weights model: the values with highest and lowest Hamming weight are placed in the extremities of the amplitude voltage, and the remaining values cluster in groups that are indicative of their Hamming weight.

For the attack phase, 1 000 traces concealing a fixed key were captured. The results of the template attacks are listed in Table 5.7. Since the template attacks run much faster, the amount of retrials and the granularity of the number of traces was increased. Even though the templates that were built in

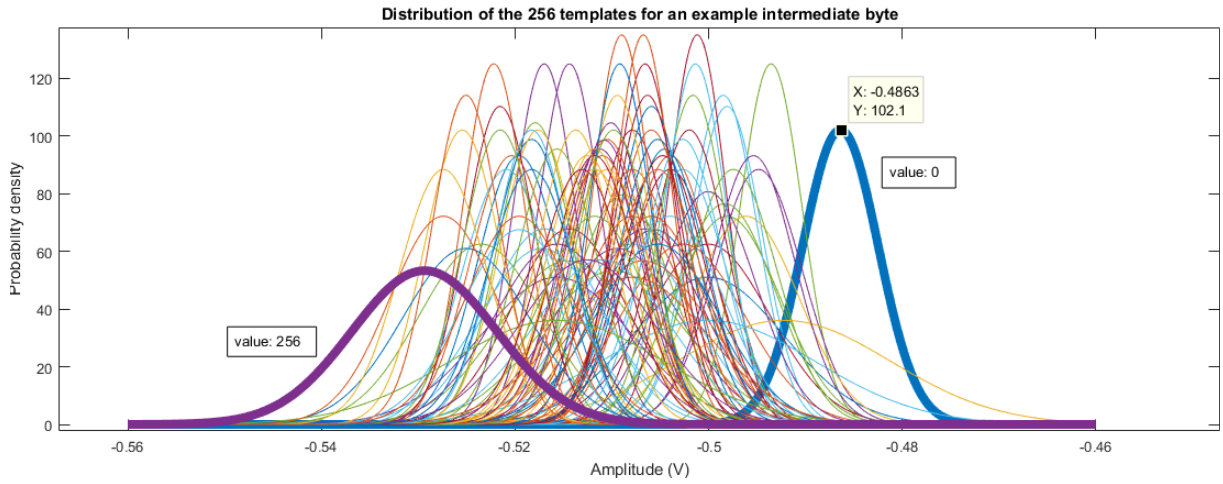

Figure 5.18: Templates for the 256 possible values of an attacked intermediate state byte.

the profiling phase had a low number of samples, the results are surprisingly positive. At a low number of traces, the template attack is notoriously more efficient: for 20 traces, the correlation attack fails to guess any complete key and only guesses the subkeys half the time; whilst the template attack guesses 11% of the complete keys, and guesses the subkeys approximately 88% of the time. The comparison is presented in Figure 5.19. For higher number of traces, both attacks display an insignificant difference between each other in all relevant metrics.

Table 5.7: Results of power univariate template attacks to the first round of AES in the smartcard using the external amplifier (averages of 100 retrials sampled from 1 000 traces)

Number of traces	SR 16 subkey average	SR^{CK} complete key	Correct normalized dist. 16 subkey average	Correct subkey position (improvement vs. correlation)
5	0.263 ± 0.011	0.000 ± 0.000	-0.081 ± 0.006	16.517 ± 0.721
10	0.598 ± 0.012	0.000 ± 0.000	1.749 ± 0.623	5.473 ± 0.328 (93.3%)
15	0.790 ± 0.010	0.000 ± 0.000	11.683 ± 9.841	2.474 ± 0.157
20	0.879 ± 0.008	0.110 ± 0.031	2.344 ± 0.329	1.536 ± 0.065 (87.2%)
25	0.939 ± 0.006	0.380 ± 0.049	2.278 ± 0.236	1.219 ± 0.052
30	0.968 ± 0.004	0.650 ± 0.048	5.721 ± 2.994	1.091 ± 0.023 (-13.7%)
35	0.989 ± 0.003	0.830 ± 0.038	3.937 ± 1.234	1.023 ± 0.007
40	0.992 ± 0.002	0.890 ± 0.031	4.622 ± 1.002	1.011 ± 0.003 (1.1%)
45	0.998 ± 0.001	0.970 ± 0.017	5.405 ± 0.974	1.002 ± 0.001
50	0.999 ± 0.001	0.980 ± 0.014	5.747 ± 0.992	1.002 ± 0.001 (0.2%)

Recall that it is not possible to compare the normalized distance between the correlation and template attacks. The template attack uses the product of likelihoods to select the most likely partial key, thus, it is much more sensitive to small perturbations than the correlation attack. As such, template attacks exhibit a much larger range of possible normalized distance values, which accounts for the increased variation in the calculation of the normalized distance.

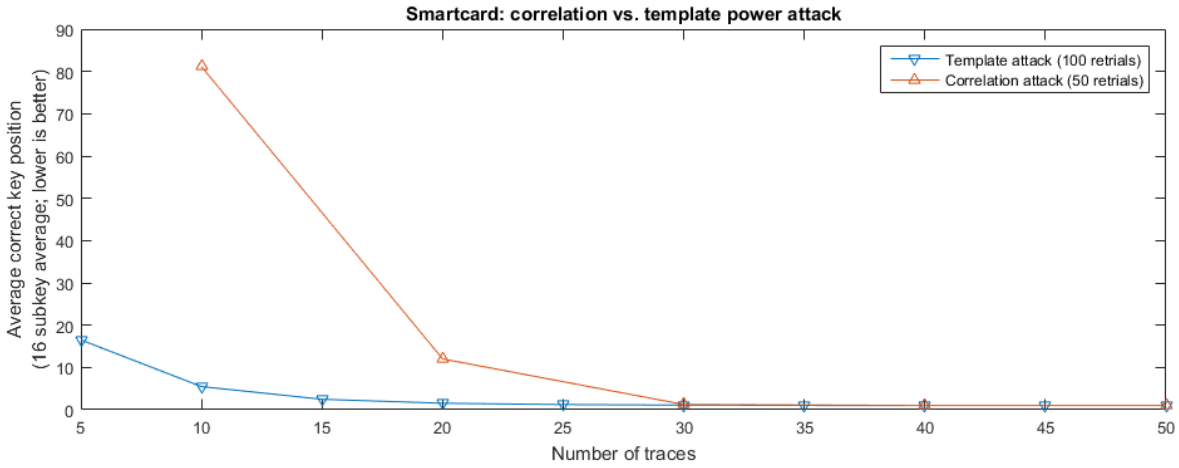

Figure 5.19: Results of the correlation and template attacks, run over different number of traces (50 retrials for correlation attacks; 100 retrials for template attacks, both selected out of 1 000 traces).

In conclusion, the template attack shows a clear and statistically significant improvement when compared to correlation attacks in smartcard. This is to be expected, since template attacks are known to be optimal in the information-theoretical sense [20].

5.4 Summary of results

The power attacks conducted successfully validated several of the techniques that can be used in EM attacks. Of the two devices, the smartcard displayed the most leakage, and is thus suited as the best candidate to test for electromagnetic attacks, considering that the smartcard not only presented the best attack results, but the signal captured had a much stronger power. In comparison, the amplified traces of the FPGA only reached the tens of millivolts in amplitude, implying that the leakage of the circuit is much inferior.

The correlation filtering technique demonstrated clear benefits in the correlation attacks and is a good candidate to improve electromagnetic attacks, which also benefit from filtering techniques.

The template attacks were very effective in attacking the smartcard, not only improving on the results of the correlation attack but converging faster to the correct results. As such, it is a promising attack to test in the electromagnetic channel.

Chapter 6

Electromagnetic attacks

In this chapter, the experimental work on electromagnetic attacks is presented, starting with a discussion on the relevant differences of the experimental setup when adapted to the EM channel, followed by the presentation of results and their discussion. Based on the previous acquired knowledge in the execution of power attacks, new experiments were carried out in the context of EM attacks in order to evaluate the success of the attacks and some signal processing techniques. Nevertheless, there are important differences in EM attacks that have to be taken into account in order to successfully exploit devices using this channel.

6.1 Experimental setup

The experimental setup shares some common features with the power attack setup. Conceptually, it takes advantage of the same framework, consisting of a device under attack, a signal acquisition device and an attacker device. In terms of equipment, the same oscilloscope was used for signal acquisition, and for the attacker device, the same computer and same software was used in order to capture, process and attack the traces. The reader is referred to the chapter 5 in order to understand the conceptual framework, the hardware available and the software developed.

There are however important differences when designing this new experimental setup. In particular, the measured signal in electromagnetic attacks is expected to have considerably less amplitude compared to the power channel. Taking into account the stronger leakage observed when attacking the power channel, the smartcard was chosen as the only device under attack for the EM attacks, in order to maximize the chances of success for this effort.

In order to capture the electromagnetic traces, it is necessary to use an appropriate probe in order to convert the EM radiation into an electrical signal that the oscilloscope can capture. Also, considering that the distance of the probe to the circuit being tested is an important factor, in order to place the probe closer to the smartcard, a *smartcard extender* was built with a small cutout which allows the probe to maintain contact with the surface of the smartcard while executing operations. In the following sections, the specific characteristics of the probe and the newly designed *smartcard extender* will be detailed.

6.1.1 Probe characteristics

The probe is responsible for the conversion of the electromagnetic radiation into an electrical signal. In this case, a near-field attack is proposed, using magnetic field probes. The Beehive Electronics 100 Series EMC probes will be used, a set of 3 magnetic field probes with different loop sizes and one electric field probe. While a bigger loop may capture a stronger signal, it will also capture neighbouring noise coming from the radiation of nearby components. The smaller loop sizes allow for increased spatial resolution and better selectivity, and as such, the small magnetic loop probe, Beehive Electronics 100B, will be the preferred choice for the attacks.

A probe amplifier is necessary to obtain traces with enough power to be captured by the acquisition device reliably. The Beehive Electronics 150A EMC probe amplifier is used to amplify the signal from the probes by 32 dB ($\pm$ 1.5 dB), in the frequency range of 100 kHz to 6 GHz, with a noise figure between 5 and 6 dB.

6.1.2 Smartcard extender

The smartcard extender is a custom-built printed circuit board that facilitates the EM attacks, depicted in Figure 6.1. The device was conceptualized by Prof. Ricardo Chaves, and later designed and built in-house at the research facility by colleague Ruben Afonso. It connects to an existing smartcard reader, and extends each of the contacts to an external card slot, creating a separation between the reader and the slot where the smartcard is placed. When a smartcard is inserted in the extended slot, a cutout allows the placement of the probe directly on top of the chip area. Due to the separation from the reader, only the EM emanations from the smartcard are captured by the probe, avoiding contamination from the nearby reader circuit.

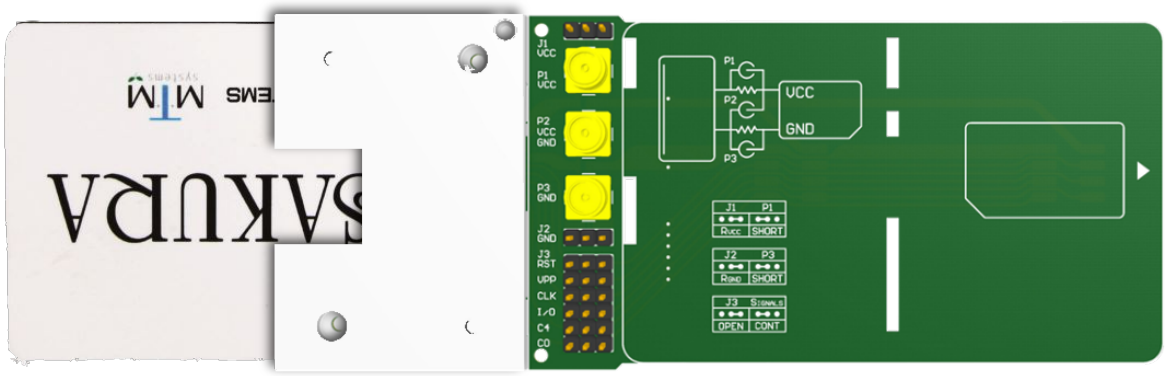

Figure 6.1: Picture of the smartcard extender. A smartcard was placed in the extended slot, and the cutout reveals the surface under which the microprocessor resides.

The smartcard extender also includes several output ports for power attacks, by including a 1 ohm resistor in series with the voltage input (VCC) and the voltage reference (GND). Jump connectors let the attacker selectively connect or disconnect each of the contacts of the smartcard. The extender can be detached into two pieces, one interfacing with the reader and the other with the smartcard. By wiring

both pieces, an attacker can potentially increase the distance between them.

An anechoic chamber was available at the laboratory, and an attempt was made to include the card slot and the probe inside the chamber in order to eliminate external sources of EM noise. Taking advantage of the ability of smartcard extender to separate into two pieces, the external slot was placed inside the chamber and the reader piece was inserted into the SAKURA-W. Unfortunately, the connectors in the chamber, which connect signals from the outside of the isolated chamber to its inside, did not have the required bandwidth to allow the smartcard to function properly. Also, the wiring between the two pieces of the extender acted as an antenna, introducing noise into the smartcard circuitry, from not only external sources but also the strong coupling between the different wires. Given the spatial resolution of the probe, which selects only the radiation in the area under the small magnetic coil, the minor expected improvements in noise reduction from external EM sources did not compensate for the significant increase in noise introduced in the smartcard by the setup, which effectively hindered its normal functioning. As such, this approach was not pursued further.

6.1.3 Probe placement

A systematic analysis of the circuit hotspots was carried out, consisting of two steps: selection of signals and verification of leakage using t-test analysis.

The smartcard was set to carry out sequential random encryptions. Meanwhile, the smallest magnetic probe was placed directly on top of the chip area (as shown in Figure 6.2) and the surface was scanned while monitoring the signal displayed by the oscilloscope on the probe channel. The smallest probe has the best space resolution since it captures the signal on a smaller area of the chip. The persistence mode was enabled at each location. This mode allows the attacker to evaluate how consistent the signal is in relation to the trigger and its variation in amplitude. One expects the leakage signal to be fairly periodic, with some relation to the clock signal, while exhibiting some data-dependent variation in amplitude. Interesting hotspots were identified and the collection of traces was carried out for the fixed vs. random t-test.

The second step, t-test analysis, was carried out to validate the existence of leakage. In a fixed vs. random t-test, both a fixed and a random set must be generated. For the random set, random subkeys and plaintexts were generated.

For the fixed set, to detect the leakage of the first round for all the 16 bytes of the state, each intermediate calculation was set to zero, in this case, the intermediate calculation is each byte of the AES state after the SubBytes operation. Random keys were generated, and the plaintext bytes were computed such that the intermediate calculation resulted in zero. Mathematically, given some randomly generated partial key K_s, the equation to solve is

$$SboxAES(P_i \oplus K_s) = 0 \Rightarrow P_i = SboxAES^{-1}(0) \oplus K_s, \tag{6.1}$$

by isolation of the plaintext byte p_i. The plaintexts generated ensure the value is fixed when combined with the key that generated them. During the capture, the random and fixed traces were randomly

Figure 6.2: Probe placed on the smartcard surface at the optimal position.

shuffled to ensure no bias from sequentially performing operations with the same fixed value.

One spot showed very good leakage, where the different operations of the first round were clearly visible, depicted in Figure 6.3. Some of the locations chosen displayed a strong signal, but in the verification step were not exhibiting significant leakage. Further analysis concluded that the intense signal observed was due to interference from the communication pins on the smartcard, which radiates considerably. This was verified by extending the duration of the capture and verifying the presence of a square wave noise that most likely was due to data transmission. As such, the attacker must be careful not to assume the correct signal is the strongest, and should always proceed with a careful analysis to verify leakage locations.

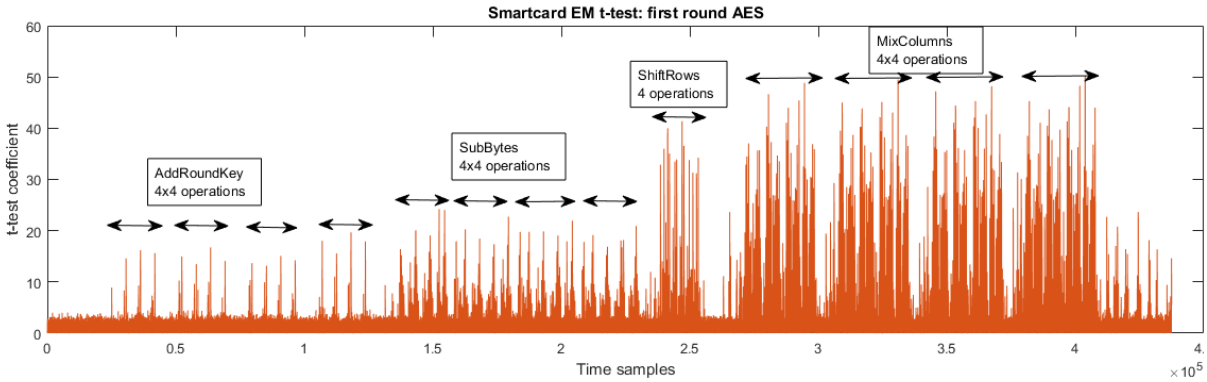

Figure 6.3: T-test trace (intermediate values fixed at zero versus random) for the EM leakage at the final probe position.

6.1.4 Complete setup

In conclusion, a final setup is presented in Figure 6.4. It is inspired in the FPGA and smartcard setups, accommodating the specific requirements of the electromagnetic attacks and taking advantage of the available resources, such as the smartcard extender.

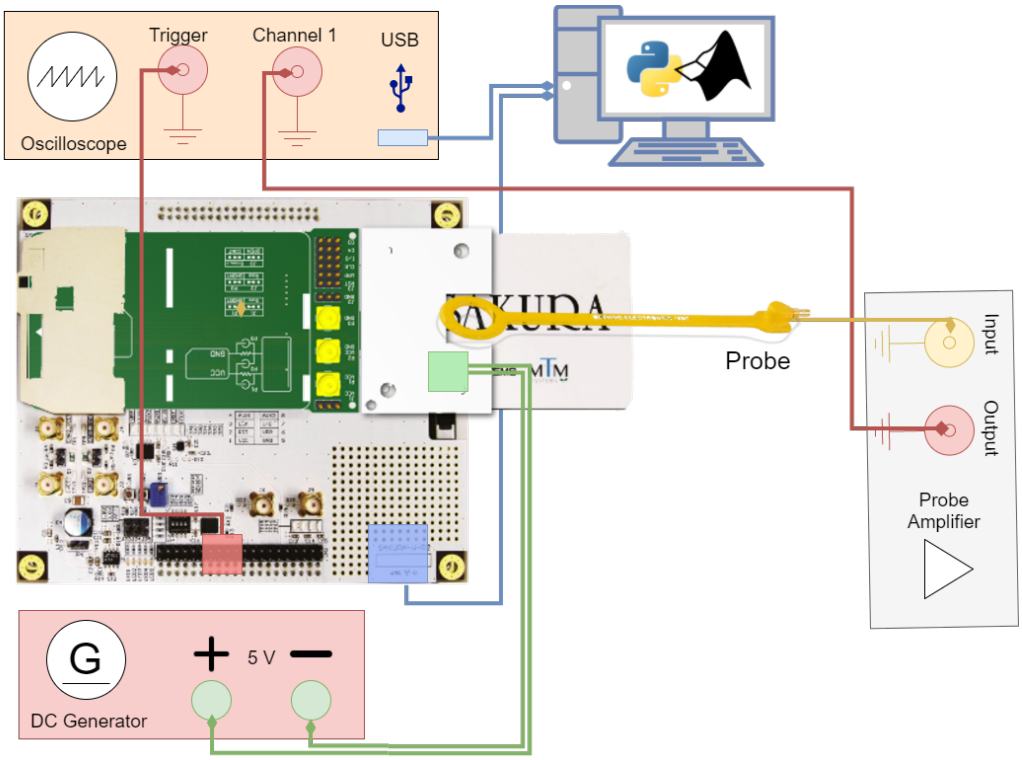

Figure 6.4: Diagram of the complete setup for the SAKURA G reference board power attack.

In an EM attack, the arrangement of the smartcard platform SAKURA-W is no different from the power attack — the attack is identical, but a different channel is being attacked. The power supply is set for a 5 V output voltage, and the board configured to use the external power source. The probe is connected to the probe amplifier using a flexible coaxial cable, to allow for better maneuverability. The probe amplifier is connected to the selected oscilloscope channel. The trigger signal is selected among the I/O pins and connected to the oscilloscope as before — throughout this thesis, the alignment of traces is always guided by the attack device, and the EM attacks are no exception. Finally, the power source provided with the probe amplifier is connected to the grid. After the board and oscilloscope are connected to the computer, the equipment is finally ready to begin an electromagnetic attack.

6.2 Smartcard Results

This section describes the results obtained for the electromagnetic attacks, using the smartcard, which was the only device tested. The techniques tested in power attacks were retested in the EM channel and their efficacy was evaluated. In general, it was possible to successfully attack the smartcard on this channel using the acquired know-how from power attacks. This is expected and in accordance to the

peer-reviewed results reported in the literature.

From the previously described setup, and using the optimal probe position, traces were captured. An example of a smartcard trace captured in the electromagnetic channel is shown in Figure 6.5. The signal captured is much weaker, on the order of millivolts, and is easily corrupted by noise. A priori, no assumptions on the frequency content are made, but it is never possible to completely capture the entire frequency content of the analog signal. Since in this case one may expect to need much more data to recover any partial keys, a compromise was struck by lowering the sampling frequency in order to save storage for capturing more traces. This limits the range of frequencies captured in other to avoid distortion, but recall that for near field attacks, such as this one, low frequency leakage is expected.

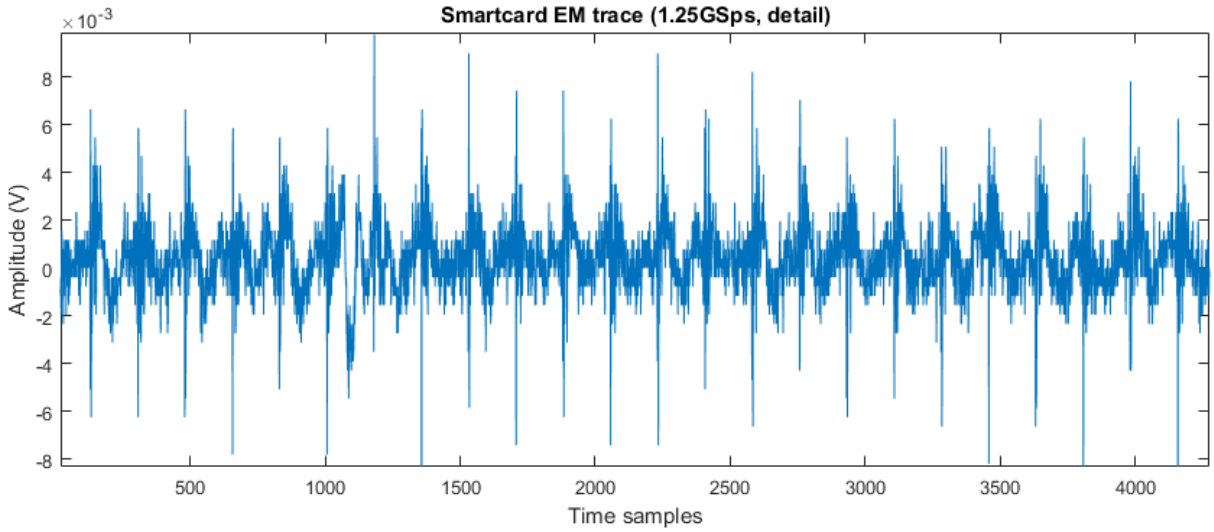

Figure 6.5: Detail of a trace captured using the EM channel at sampling frequency of 1.25 GHz.

6.2.1 Correlation attacks: model comparison

For the correlation attacks of the base setup, two main experiments were conducted: an experiment using Hamming Weights and an experiment using a power model named Switching Distance, which takes into account the fact that current direction affects the voltage polarity of an electromagnetic trace, the switching distance model takes advantage of this fact to classify each transition independently.

Recall that this is identical to a Hamming distance model, but the transition of $0 \rightarrow 1$ is considered a positive transition, and a transition of $1 \rightarrow 0$ is considered a negative transition and it decrements the Switching distance by one.

As in power attacks, the target is the output of the SBOX in the first round. To use the Switching distance model, it is necessary to determine the previous result from which the target value commutes. In order to establish the correct previous value, the Switching distance was evaluated as the transition between the target value (the SBOX output), and, alternatively, either the key addition, the plaintext byte or the key byte. A systematic evaluation determined that attacks were only successful for the distance between the SBOX output and the partial key, which suggests the registers used to store the SBOX output of the first round were previously used to store the key.

The comparison of both models is illustrated in Figure 6.6. There is a minor performance advantage for the attacks using the Switching distance, but it is not significant. The Hamming Weights model performs almost identically to the Switching distance for all the tested attacks, especially for attacks using a substantial number of traces, which lead to similar results in both cases. There is a small improvement in the attacks with lower number of traces, but it is not statistically significant for the number of retrials tested.

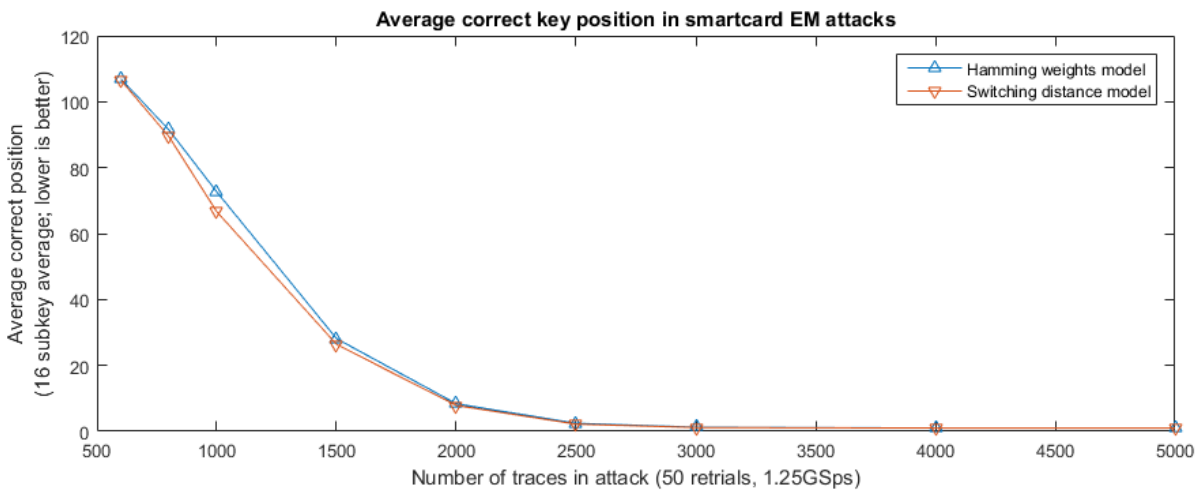

Figure 6.6: Comparison between a Hamming Weights and Switching Distance model in electromagnetic attacks to a smartcard (50 retrials).

In summary, albeit the two models tested present very different power hypotheses, both correlation attacks using either model presented similar results, with a minor advantage to the Switching distance model. The experimental work will proceed using the Hamming Weights model, as in other previous experiments.

Correlation attacks: correlation filtering

The usage of correlation filtering was tested in electromagnetic correlation attacks, taking into account the promising results of this technique in the power attacks. The typical procedure was followed, starting with a cutoff frequency sweep. The result of the frequency sweep is illustrated in Figure 6.7. A clear improvement is noticeable for filters with cutoff frequencies above normalized frequency 0.025 (corresponding to a true frequency of $0.025 \times 1250 \div 2 = 15.625$ MHz).

A zoomed version of the same sweep is presented in Figure 6.8. When before an attack of 800 traces lead to an average correct position of approximately 90, this metric is now just under 2. This suggests most of the noise is high-frequency (higher than the range swept) and that the main frequency content related to leakage is located under 12.5 MHz, justifying the catastrophic loss of success for filters with lower cutoff frequencies. The filter with cutoff frequency of 0.026 in normalized units was selected as the local minima in the weakest attack using 800 traces. According to the normalization used, this low pass filter has a true cutoff frequency of 16.25 MHz.

For positive results like these, a complete run for different number of traces was run with the optimal

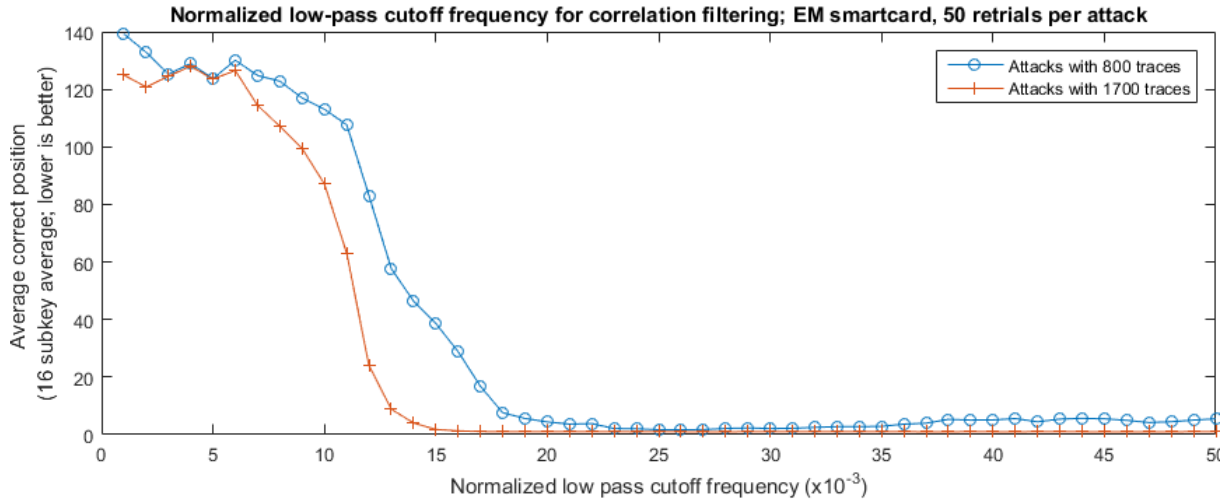

Figure 6.7: Cutoff frequency sweep for correlation filters (50 retrials, 1.25GSps, in normalized frequencies, true frequency sweep from 0.625 MHz to 31.25 MHz).

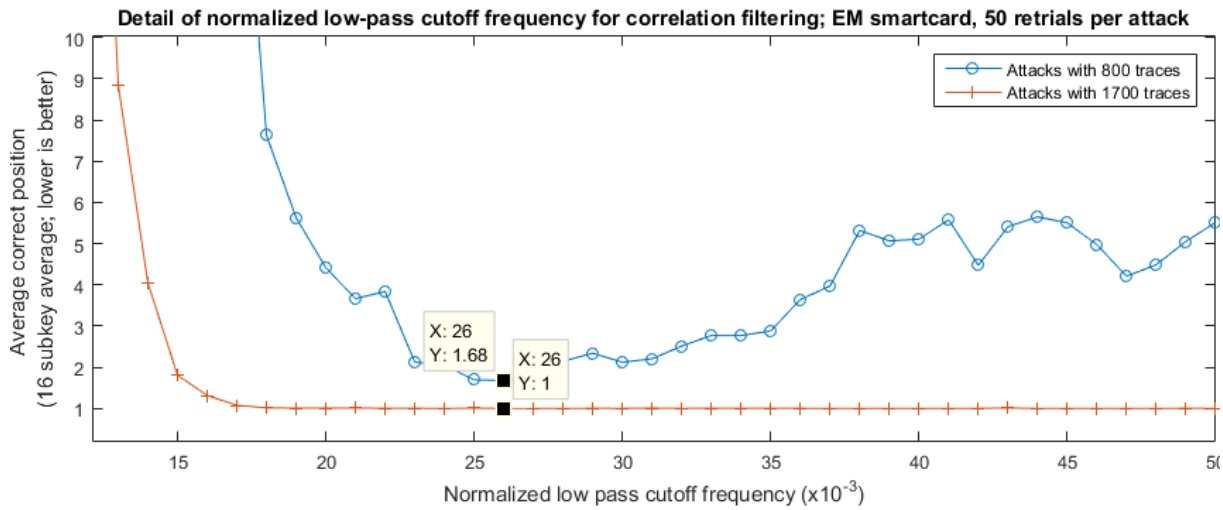

Figure 6.8: Detail of the cutoff frequency sweep for correlation in relevant area (50 retrials, 1.25GSps, true frequency from 0.625 MHz to 31.25 MHz).

cutoff frequency, to compare with the original results. The improved results are shown in Table 6.1. A comparison is shown in Figure 6.9 for the average correct key position, and in Figure 6.10 for the correct normalized distance.

The attacks using filtered correlation not only display improved metrics (175% higher average subkey success rate) but also decay exponentially faster regarding the average correct key, and increase faster when considering the normalized distance. When taking into account the success rate of complete keys, it takes less than half the traces (2000 vs. 5000) to reach approximately complete success (1.0).

Table 6.1: Results of electromagnetic smartcard correlation attacks to the first round of AES, using the probe amplifier and correlation filtering with -6dB cutoff at 16.25 MHz, with the Hamming weights model (averages of 50 retrials sampled from 5 500 traces)

Number of traces	SR 16 subkey average	SR^{CK} complete key	Normalized dist. 16 subkey average	Correct subkey position (improvement vs. HW)
100	0.03 ± 0.01	0.00 ± 0.00	-0.17 ± 0.00	111.50 ± 2.65
200	0.09 ± 0.01	0.00 ± 0.00	-0.12 ± 0.01	85.63 ± 2.56
300	0.18 ± 0.01	0.00 ± 0.00	-0.06 ± 0.01	50.40 ± 2.20
400	0.34 ± 0.02	0.00 ± 0.00	0.02 ± 0.01	24.11 ± 1.48
500	0.53 ± 0.02	0.00 ± 0.00	0.08 ± 0.01	10.12 ± 0.83
600	0.67 ± 0.02	0.00 ± 0.00	0.16 ± 0.01	4.85 ± 0.49 (95.5%)
700	0.80 ± 0.01	0.02 ± 0.02	0.21 ± 0.01	2.22 ± 0.24
800	0.88 ± 0.01	0.02 ± 0.02	0.27 ± 0.01	1.34 ± 0.05 (98.5%)
1000	0.97 ± 0.01	0.64 ± 0.07	0.38 ± 0.01	1.05 ± 0.02 (98.6%)
1500	0.99 ± 0.00	0.90 ± 0.04	0.56 ± 0.02	1.01 ± 0.00 (96.4%)
2000	1.00 ± 0.00	0.98 ± 0.02	0.70 ± 0.02	1.00 ± 0.00 (88.0%)

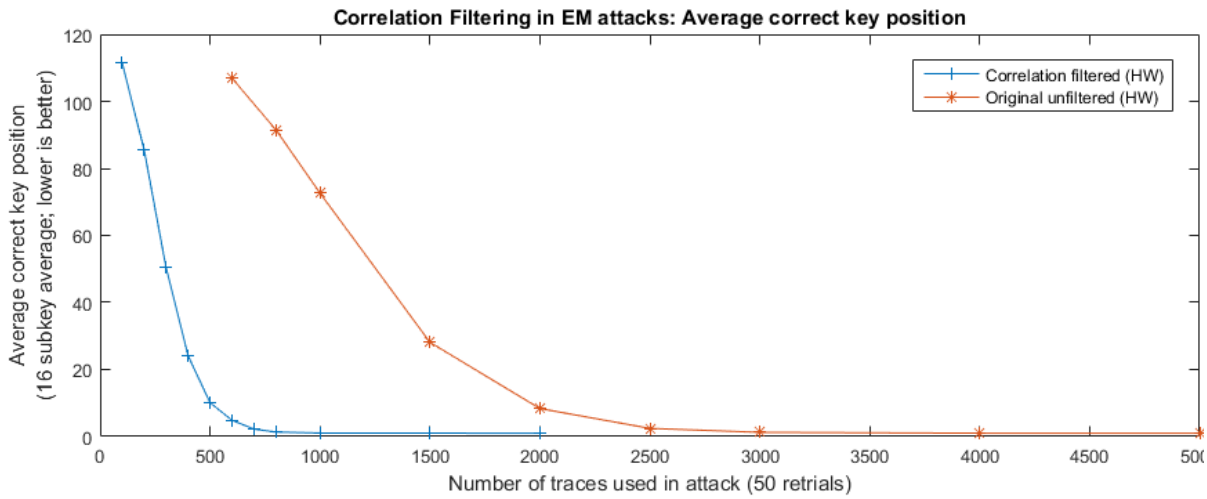

Figure 6.9: Results of average correct partial key position for EM correlation attacks with different number of traces, with and without correlation filtering, for 50 retrials using the Hamming weights model.

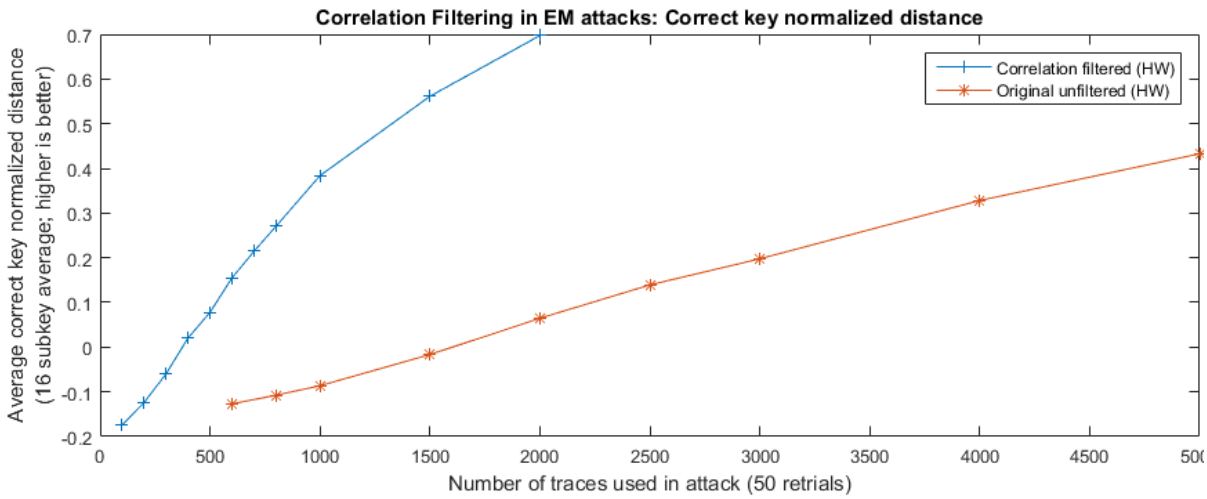

Figure 6.10: Results of average correct normalized distance for EM correlation attacks with different number of traces, with and without correlation filtering, for 50 retrials using the Hamming weights model.

To conclude, the success of the correlation filtering technique was replicated in the electromagnetic channel. The improvement observed, when compared to unprocessed attacks, is even greater than in the power attack case. This is possibly due to the more important role of noise in these attacks, which the correlation filtering helps address.

6.2.2 Correlation attacks: setup comparison

A last effort was made to improve the results, by additionally amplifying the signal, combining the external amplifier with the probe amplifier. When the dual amplifier approach was tested in the FPGA attacks, it did not lead to improved results. However, unlike in the FPGA case, the signal captured is far from occupying the entire range of the oscilloscope, which is set to the smallest range, from -50 mV to 50 mV. By amplifying the signal and setting an adequate DC offset in the oscilloscope, it is possible to reduce the quantization error by taking advantage of the 8 bits divided in that range, by filling the range of the oscilloscope with the probe signal. The external amplifier was set at 9.75V, which did not correspond to any gain curve in the specification of the amplifier. However, taking into account the previously acquired samples without an external amplifier, it is possible to calculate the gain G from the ratio of amplitudes between the signals before and after external amplification (A_{start} and A_{end}, respectively):

$$G = 20 \cdot \log_{10} \left(\frac{A_{end}}{A_{start}} \right) \text{ [dB]} \tag{6.2}$$

Thus, by comparing the amplitudes of the signals and their power, the gain is calculated using the aforementioned formula to be approximately 18 dB, which is coherent with the nearest data sheet curve for 12 V (with typical gain of 20 dB).

In order to provide a consistent baseline, no signal processing techniques were applied. First, 5 500 traces were captured at the same sampling frequency of 1.25 GHz. Correlation attacks to the first round were carried out using the Hamming Weights model, and the results are displayed in Table 6.2.

Table 6.2: Results of electromagnetic smartcard correlation attacks to the first round of AES, using the probe amplifier and external amplifier, with the Hamming weights model (averages of 50 retrials sampled from 5 500 traces)

Number of traces	SR 16 subkey average	SR^{CK} complete key	Normalized distance 16 subkey average	Correct subkey position (improvement vs. HW)
600	0.07 ± 0.01	0.00 ± 0.00	-0.11 ± 0.00	92.85 ± 2.59 (13.15%)
800	0.11 ± 0.01	0.00 ± 0.00	-0.08 ± 0.00	64.38 ± 2.26 (29.61%)
1000	0.20 ± 0.01	0.00 ± 0.00	-0.04 ± 0.00	40.55 ± 1.88 (44.18%)
1500	0.52 ± 0.02	0.00 ± 0.00	0.06 ± 0.01	6.98 ± 0.56 (75.17%)
2000	0.83 ± 0.01	0.08 ± 0.04	0.15 ± 0.01	1.60 ± 0.09 (80.90%)
2500	0.97 ± 0.01	0.66 ± 0.07	0.24 ± 0.01	1.03 ± 0.01 (57.63%)
3000	1.00 ± 0.00	0.94 ± 0.03	0.33 ± 0.01	1.00 ± 0.00 (20.24%)
4000	1.00 ± 0.00	1.00 ± 0.00	0.48 ± 0.01	1.00 ± 0.00 (0.25%)
5000	1.00 ± 0.00	1.00 ± 0.00	0.59 ± 0.01	1.00 ± 0.00 (0.00%)

The comparison between the original setup and the setup with the external amplifier is illustrated in Figures 6.11 and 6.12, for the average correct key position and normalized distance, respectively.

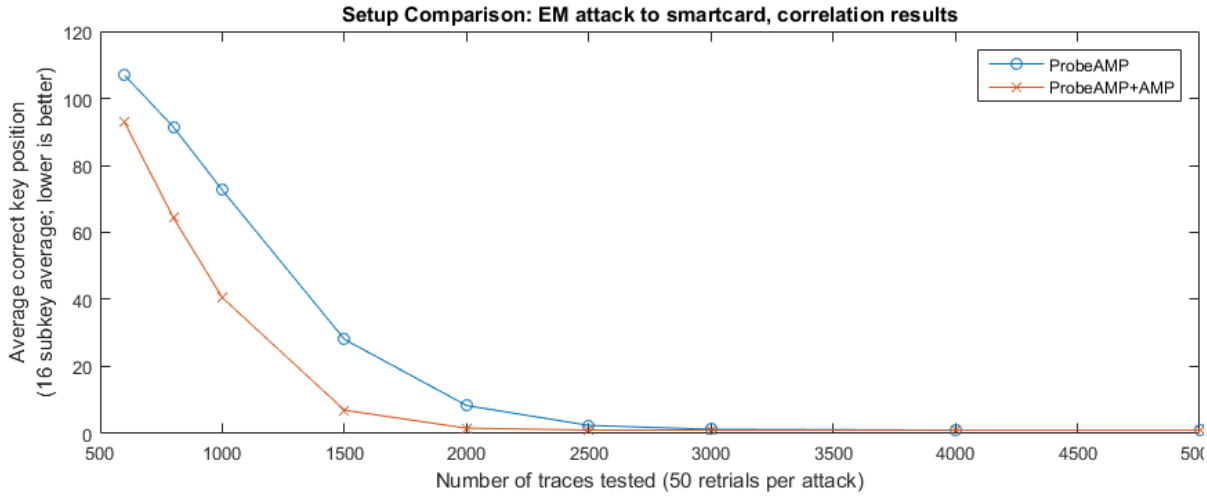

Figure 6.11: Results of average correct partial key position for EM correlation attacks with different number of traces, with and without additional external 18dB amplification, for 50 retrials using the Hamming weights model.

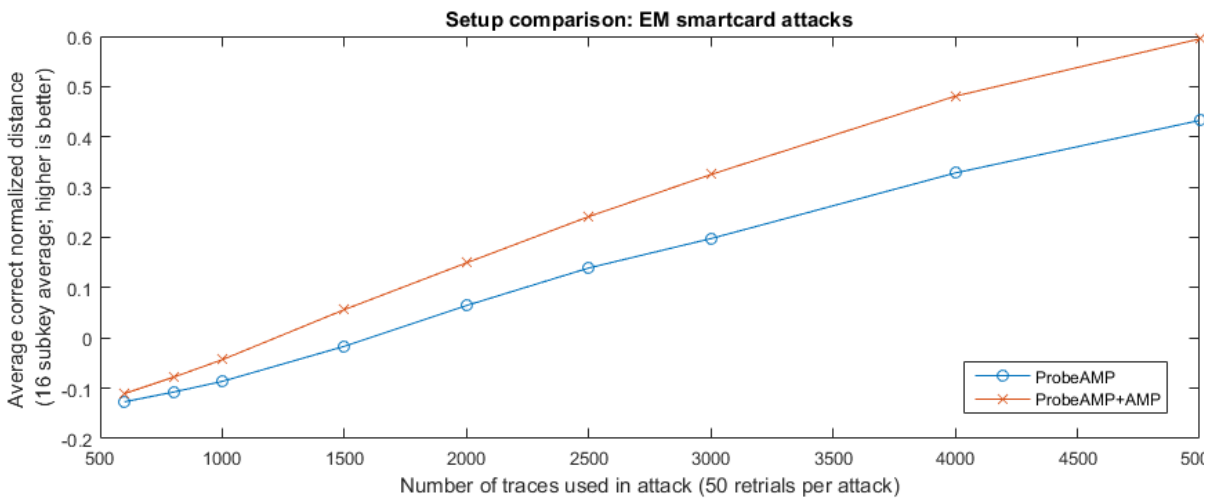

Figure 6.12: Results of average correct normalized distance for EM correlation attacks with different number of traces, with and without additional external 18dB amplification, for 50 retrials using the Hamming weights model.

The improvements in the results are significant, exhibiting a clear advantage in the additional amplification. For the average correct position in Figure 6.11, there is an overall improvement between 10 and 30% across all attacks with different number of traces, however, the decay rate is similar in both cases. The normalized distance in Figure 6.12 further confirms this improvement.

In conclusion, the implementation of a setup with further amplification is beneficial for the EM attacks, especially taking into account the reduction of quantization noise due to filling the range selected for the oscilloscope.

6.2.3 Template attacks

Univariate template attacks showed promising results in the smartcard power attacks and, as such, they too were tested in the electromagnetic channel. While these attacks did attain some measure of success,

recovering correct keys some percentage of the time, the template attacks failed to outperform the correlation attacks. What follows is the analysis and interpretation of the results of the electromagnetic template attacks in light of the side-channel theory previously presented.

Templates were built using an optimized procedure, similar to the one used for conducting the t-test. Traditionally, generating random keys and plaintexts could result in an uneven distribution of samples for each intermediate value, especially if the number of traces is not very large. Considering the attack targets the first round after the SBOX, it is possible to freeze each value of the SBOX output, using the same formula as presented in Equation 6.1 but replacing the zero with the intended fixed value.

For example, if 30 instances of an intermediate value are fixed by calculating the corresponding plaintext and key pair, this will result in 30 samples captured for that value. In the case of AES, the attack targets each byte individually, then, $30 \times 256 = 7\ 680$ traces would be needed. The plaintext and key pairs are randomly sorted to ensure that there are no clear biases when capturing the traces. This was the procedure followed, but since traces were captured in blocks of 100, not all traces were captured. The final templates used 7 600 traces (80 traces missing), having between 25 and 30 samples per intermediate value, with an average of 29.69 samples per value. The templates were captured for the points of interest that showed greatest leakage in the previous correlation attacks.

For the first byte of the intermediate state, the templates are represented in Figure 6.13. A simple observation of the templates leads to the conclusion that they are, in fact, much less separated than the equivalent templates in the power analysis attack. This suggests the attacks will require more traces, as confirmed in the correlation attacks.

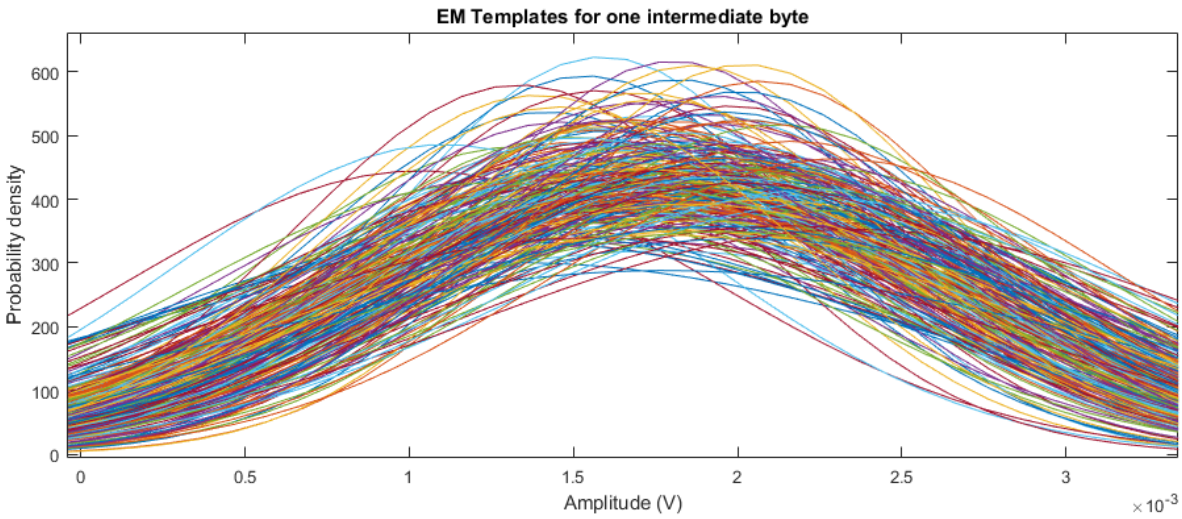

Figure 6.13: EM templates built for one of the intermediate state bytes.

The online phase of the attack was carried out against the same set of traces as the previous correlation attack, without external amplification. This provides the baseline for the comparison analysis. An attentive eye may notice that the template attacks were sampled from 10 000 traces, instead of 5 500. In fact, these traces were all captured in the same session as those in the correlation attack, and the total number of traces was only increased in order to test attacks that utilized more traces. The results of the template attacks are listed in Table 6.3.

Table 6.3: Results of electromagnetic smartcard template attacks using the probe amplifier (averages of 100 retrials sampled from 10 000 traces)

Number of traces	SR 16 subkey average	SR^{CK} complete key	Normalized distance 16 subkey average	Correct subkey position (improvement vs. HW)
1000	0.128 ± 0.008	0.000 ± 0.000	-0.002 ± 0.000	41.824 ± 1.296 (42.4%)
2000	0.255 ± 0.011	0.000 ± 0.000	-0.001 ± 0.000	25.268 ± 1.049 (-202.2%)
3000	0.369 ± 0.012	0.000 ± 0.000	-0.000 ± 0.000	17.116 ± 0.918 (-1258%)
4000	0.492 ± 0.013	0.000 ± 0.000	0.000 ± 0.000	11.933 ± 0.763 (-1193%)
4500	0.517 ± 0.012	0.000 ± 0.000	0.000 ± 0.000	10.957 ± 0.747
5000	0.567 ± 0.012	0.000 ± 0.000	0.000 ± 0.000	10.569 ± 0.764 (-1056%)
5500	0.633 ± 0.012	0.000 ± 0.000	0.001 ± 0.000	9.554 ± 0.744
7000	0.737 ± 0.011	0.000 ± 0.000	0.001 ± 0.000	8.606 ± 0.725
7500	0.776 ± 0.010	0.000 ± 0.000	0.001 ± 0.000	7.319 ± 0.631
8000	0.799 ± 0.010	0.000 ± 0.000	0.001 ± 0.000	7.951 ± 0.689

The comparison presented in Figure 6.14 suggests that the performance of the template attacks is worse than the performance of correlation attacks. Template attacks are claimed to be *"the strongest form of side channel attack possible in an information theoretic sense"* [20], but inevitably depend on the model built with the templates and the points of interest that are used.

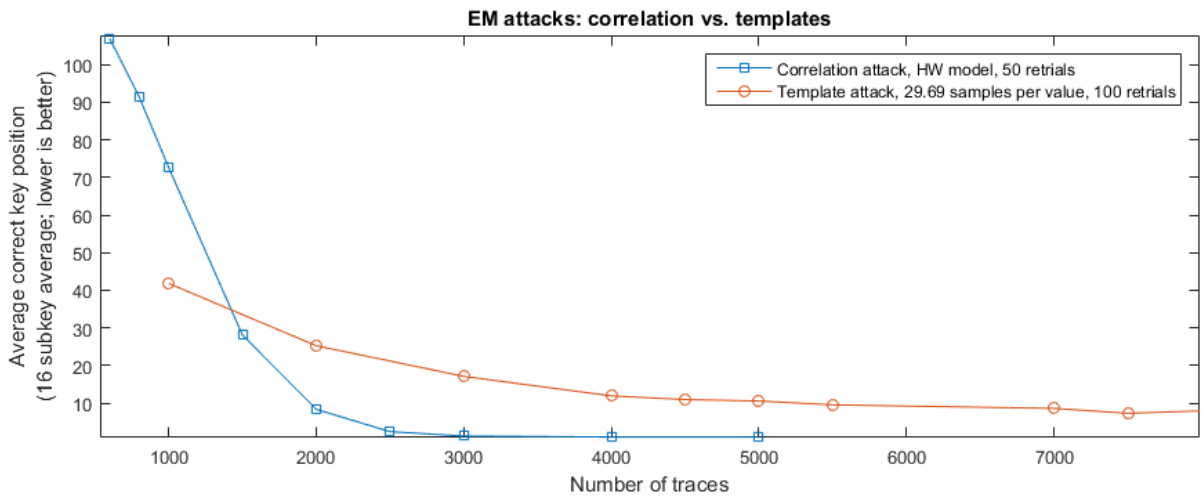

Figure 6.14: Results for the EM template and correlation attacks, compared.

Regarding the templates built with 7600 traces (average of 29.69 samples per intermediate value), the total number of traces used was much greater than the number of traces used in any of the correlation attacks. There is of course the possibility of human error, perhaps by shifting the probe imperceptibly from the optimal position during the handling of the setup at the laboratory. However, successful correlation attacks with similar metrics to those in the previous experiments were possible with traces captured *after* the traces used in the template, which suggests that this was not a likely cause. Templates were generated with 50 samples per intermediate value and retested, but the results of small-scale testing did not improve, in fact, they were slightly worse.

To further increase the number of samples for each intermediate value, it is necessary to collect additional traces. In that case, if all the time samples are collected — perhaps to allow the attacker to change points of interest, another factor which may affect the performance of the templates, — the

amount of traces needed would surpass the tens of gigabytes. Given the circumstances, it was deemed unpractical to rerun the profiling phase.

To test the mechanism that generates the templates, instead of using fixed values to generate the templates, the traces in the random portion of the t-test were used instead. While these traces clearly gave positive leakage results in the t-test analysis, they did not improve the results of the template attacks.

Regarding the points of interest, several legitimate selections were tested. Initially, the points of interest selected were based on the highest leakage time samples in the correlation attack, but other possible leakage points were retrieved from the t-test as possible points of interest. In any case, in small testing with single attacks, no discernible improvement in results was observed.

To conclude, template attacks are very sensitive to the templates used, allowing the attack to generally extract more information from the samples, but at the expense of easily failing when the model is not adequate. Differences between the setup of training and the attack setup can bring worse results than a naive model informed by the theory. It is possible that the bad results observed are simply due to the templates not having enough samples, since the traces used have low amplitude and are thus affected by a significant amount of noise. The visual inspection of the templates corroborates this hypothesis, due to poor separation of the values.

Chapter 7

Conclusion

The security of electronic devices is paramount to our information society. The majority of the devices already implement standardized secure algorithms, but suffer from implementation issues. Side-channel attacks enable a potential attacker to stealthily compromise the devices in question. In particular, electromagnetic side-channel attacks can be carried out without physically tampering the device, and are thus particularly hard to detect and prevent.

The findings promise to introduce great value to the cryptanalytic community, as they consolidate the different signal processing techniques used in the field, assess their performance in electromagnetic side-channel attacks, and enable direct, consistent comparisons among them. This work is focused mainly on two attack techniques: correlation attacks and template attacks. To study these techniques, experimental work was first carried out as power consumption attacks. Later, having acquired know-how, the attacks were carried out using the electromagnetic emanations. The achieved objective of evaluating these attacks enables now their direct comparison using a well-defined methodology of analysis.

In correlation attacks, two signal processing techniques were studied: trace filtering and correlation filtering, a proposed technique based on empirical observation of the behavior of the correlation attacks. In trace filtering, the input traces of the attack are filtered so that their frequency content is limited to some maximum frequency. On the other hand, in correlation filtering, the input traces are not processed, instead, the correlation traces are processed. During the attack, as the correlation traces are calculated, they are filtered to eliminate the sharp fluctuations of hypotheses, and instead favor the clusters of increased correlation.

Regarding trace filtering, this method provided modest results in power attacks, showing small improvements. In any case, the attack is relatively impervious to the elimination of most of its frequency content, which suggests most of the high frequencies which were removed are not relevant to the attack. This is in accordance with the existing literature in power attacks, due to the slow transitions of the registers when compared to the circuit's clock signal.

In the case of correlation filtering, the results were substantially positive, providing a significant advantage in both power and electromagnetic attacks. Provided the attacker can estimate which is the ideal filter, e.g. using a clone of the device under attack, the correlation filtering attack provided sta-

tistically significant improvements when tested, reducing the number of traces necessary to output the correct key.

The template attacks were also evaluated, but their performance was not as consistent. In the power attack case, the attack succeeded in outperforming the correlation attack, but failed in the electromagnetic attacks. Using the available data, it is possible that the results are due to inadequacy of the templates used for the electromagnetic attacks. Further testing is warranted, including development of the multivariate attacks.

In summary, the evaluation of existing side-channel techniques was successfully carried out, having extracted secret keys from the devices using the types of electromagnetic attacks studied. The work presented allows the direct comparison of these different attacks using a detailed framework, and introduces newly designed signal processing techniques which improve the success of the electromagnetic attacks.

Chapter 8

Future work

The work of this thesis is focused in addressing the objectives set forth in the Introduction and, as this dissertation progressed, relevant open areas of work became evident. These include different experimental configurations, novel side-channel techniques published in the literature, different signal processing strategies, and addressing the newer defense mechanisms that have been made available in recent years.

First, some additional experimental configurations are proposed for future work. In the experimental setup of the electromagnetic attacks, the characterization of the spatial positioning of the probe is proposed. In this thesis, the probe position was derived using a manual systematic scan of the surface of the smartcard, however, an automated scan with computer-assisted positioning would enable an attacker to profile the locations of the smartcard where the leakage is optimal.

In the context of trace alignment, the traces captured were aligned using a trigger signalled by the device under attack. In an adversarial context, such a mechanism is unlikely to be available, and as such, the development of an alignment technique that does not depend on external triggering is proposed. There are several well-known techniques that can enable this, such as cross-correlation, and the development of software to take advantage of such techniques would enable their application under an experimental setting.

Regarding template models, the importance of building reliable templates is paramount to ensure the success of template attacks. In real-world attacks, often the templates are built in a setting different from the location of the device under attack. When compared to the templates, the differences of the traces, such as a different signal power or DC offset, may hinder the template attack. One of the mechanisms proposed to facilitate the reusage of templates that are linearly transformed is the so called *template portability* concept, introduced by Elaabid et al. [22]. A review of template portability is proposed, in order to evaluate its suitability and possible improvement in template attacks. Additionally, the implementation of the multivariate template attack is proposed.

In recent years, new side-channel techniques have been published inspired by machine learning. There are several techniques in this area that are able to do classification of hypotheses and to learn power models, which promise to bring improvements to the attacks. For future work, the characterization

of these new techniques and application of signal processing tools is proposed.

Finally, there are several countermeasures developed to minimize the impact of side-channel attacks in commercial devices. Some of these countermeasures, such as masking, have known attack strategies which can be used to overcome them. The evaluation of these techniques in electromagnetic attacks, and the application of relevant signal processing techniques, would be a valuable contribution.

Bibliography

[1] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Annual International Cryptology Conference*, pages 388–397. Springer, 1999.

[2] C. E. Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4): 656–715, 1949.

[3] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[4] C. Swenson. *Modern cryptanalysis: techniques for advanced code breaking*. John Wiley & Sons, 2008.

[5] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984. ISSN 0022-0000. doi: http://dx.doi.org/10.1016/0022-0000(84)90070-9. URL `http://www.sciencedirect.com/science/article/pii/0022000084900709`.

[6] K. Sako. *Semantic Security in Encyclopedia of Cryptography and Security*, pages 1176–1177. Springer US, Boston, MA, 2011. ISBN 978-1-4419-5906-5. doi: 10.1007/978-1-4419-5906-5_23. URL `http://dx.doi.org/10.1007/978-1-4419-5906-5_23`.

[7] J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC press, 2014.

[8] D. Hans and K. Helmut. Introduction to cryptography–principles and applications, 2007.

[9] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[10] F.-X. Standaert. Introduction to side-channel attacks. In *Secure Integrated Circuits and Systems*, pages 27–42. Springer, 2010.

[11] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. *The EM Side—Channel(s)*, pages 29–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-36400-9. doi: 10.1007/3-540-36400-5_4. URL `http://dx.doi.org/10.1007/3-540-36400-5_4`.

[12] E. Peeters, F.-X. Standaert, and J.-J. Quisquater. Power and electromagnetic analysis: Improved model, consequences and comparisons. *Integration, the VLSI Journal*, 40(1):52 – 60, 2007. ISSN 0167-9260. doi: https://doi.org/10.1016/j.vlsi.2005.12.013. URL `http://www.sciencedirect.com/science/article/pii/S0167926005000647`. Embedded Cryptographic Hardware.

[13] M. Aigner and E. Oswald. Power analysis tutorial. Technical report, Institute for Applied Information Processing and Communication, Graz, Austria, 2000. URL https://www.iaik.tugraz.at/content/research/implementation_attacks/introduction_to_impa/dpa_tutorial.pdf.

[14] N. F. Pub. 197: Advanced encryption standard. *Federal Information Processing Standards Publication*, 197:441–0311, 2001.

[15] D. Genkin, A. Shamir, and E. Tromer. Rsa key extraction via low-bandwidth acoustic cryptanalysis. In *International Cryptology Conference*, pages 444–461. Springer, 2014.

[16] S. P. Skorobogatov. *Semi-invasive attacks: a new approach to hardware security analysis*. PhD thesis, University of Cambridge, 2005.

[17] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Comput.*, 51(5):541–552, May 2002. ISSN 0018-9340. doi: 10.1109/TC.2002.1004593. URL http://dx.doi.org/10.1109/TC.2002.1004593.

[18] E. Brier, C. Clavier, and F. Olivier. *Correlation Power Analysis with a Leakage Model*, pages 16–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-28632-5. doi: 10.1007/978-3-540-28632-5_2. URL http://dx.doi.org/10.1007/978-3-540-28632-5_2.

[19] P. N. Fahn and P. K. Pearson. *IPA: A New Class of Power Attacks*, pages 173–186. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. ISBN 978-3-540-48059-4. doi: 10.1007/3-540-48059-5_16. URL http://dx.doi.org/10.1007/3-540-48059-5_16.

[20] S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, pages 13–28, London, UK, UK, 2003. Springer-Verlag. ISBN 3-540-00409-2. URL http://dl.acm.org/citation.cfm?id=648255.752740.

[21] B. Gierlichs, K. Lemke-Rust, and C. Paar. *Templates vs. Stochastic Methods*, pages 15–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-46561-4. doi: 10.1007/11894063_2. URL http://dx.doi.org/10.1007/11894063_2.

[22] M. A. Elaabid, S. Guilley, and P. Hoogvorst. Template attacks with a power model. *IACR Cryptology ePrint Archive*, 2007:443, 2007.

[23] E. Oswald and S. Mangard. *Template Attacks on Masking—Resistance Is Futile*, pages 243–256. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-69328-4. doi: 10.1007/11967668_16. URL http://dx.doi.org/10.1007/11967668_16.

[24] C. Rechberger and E. Oswald. *Practical Template Attacks*, pages 440–456. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-31815-6. doi: 10.1007/978-3-540-31815-6_35. URL http://dx.doi.org/10.1007/978-3-540-31815-6_35.

[25] F. Durvaux and F.-X. Standaert. *From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces*, pages 240–262. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-49890-3. doi: 10.1007/978-3-662-49890-3_10. URL http://dx.doi.org/10.1007/978-3-662-49890-3_10.

[26] F.-X. Standaert, T. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Eurocrypt*, volume 5479, pages 443–461. Springer, 2009.

[27] B. Köpf and D. Basin. An information-theoretic model for adaptive side-channel attacks. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 286–296. ACM, 2007.

[28] O. Choudary and M. G. Kuhn. *Efficient Template Attacks*, pages 253–270. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08302-5. doi: 10.1007/978-3-319-08302-5_17. URL https://doi.org/10.1007/978-3-319-08302-5_17.

[29] J. L. Massey. Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory*, pages 204–, Jun 1994. doi: 10.1109/ISIT.1994.394764.

[30] A. D. Santis, A. G. Gaggia, and U. Vaccaro. Bounds on entropy in a guessing game. *IEEE Transactions on Information Theory*, 47(1):468–473, Jan 2001. ISSN 0018-9448. doi: 10.1109/18.904564.

[31] C. Whitnall and E. Oswald. *A Comprehensive Evaluation of Mutual Information Analysis Using a Fair Evaluation Framework*, pages 316–334. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-22792-9. doi: 10.1007/978-3-642-22792-9_18. URL https://doi.org/10.1007/978-3-642-22792-9_18.

[32] A. Thillard, E. Prouff, and T. Roche. *Success through Confidence: Evaluating the Effectiveness of a Side-Channel Attack*, pages 21–36. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-40349-1. doi: 10.1007/978-3-642-40349-1_2. URL https://doi.org/10.1007/978-3-642-40349-1_2.

[33] Y. Oren, O. Weisse, and A. Wool. Practical template-algebraic side channel attacks with extremely low data complexity. In *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, pages 7:1–7:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2118-1. doi: 10.1145/2487726.2487733. URL http://doi.acm.org/10.1145/2487726.2487733.

[34] M. S. E. Mohamed, S. Bulygin, M. Zohner, A. Heuser, M. Walter, and J. Buchmann. Improved algebraic side-channel attack on aes. *Journal of Cryptographic Engineering*, 3(3):139–156, Sep 2013. ISSN 2190-8516. doi: 10.1007/s13389-013-0059-1. URL https://doi.org/10.1007/s13389-013-0059-1.

[35] A. Heuser, O. Rioul, and S. Guilley. A theoretical study of kolmogorov-smirnov distinguishers. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 9–28. Springer, 2014.

[36] J. F. C. Lopes. Power analysis: From theory to practice. Master's thesis, Instituto Superior Técnico, Universidade de Lisboa, 2017.

[37] H. Guntur, J. Ishii, and A. Satoh. Side-channel attack user reference architecture board sakura-g. In *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, pages 271–274, Oct 2014. doi: 10.1109/GCCE.2014.7031104.

[38] M. Matsubayashi and A. Satoh. Side-channel attack user reference architecture board sakura-w for security evaluation of ic card. In *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*, pages 565–569, Oct 2015. doi: 10.1109/GCCE.2015.7398507.