POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Performing SCA Against Block Ciphers Using Closest and Furthest Leakage Models

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Costantino Vincifori**

Student ID: 918759
Advisor: Prof. Gerardo Pelosi
Co-advisors: Prof. Alessandro Barenghi
Academic Year: 2020-21

Abstract

Nowadays we are experiencing a wide-spreading of embedded systems, sensors and integrated circuits. Such device have to operate in hostile environments that might affect their intended functionality. Cryptography is one of the most common tools used to safeguard the correct functionality of such devices.

However, cryptographic implementations on physical systems are subject to *side-channel* attacks, that are attacks that exploit physical properties of the targeted device. Such physical properties are observed while the device is computing encrypting/decrypting operations. The best attacks that can be employed define a *leakage model* as the most accurate relationship that links both secret key used in a cryptographic operation and the physical property that it is taken into account.

When the leakage model cannot be defined as accurately as intended due to a partial knowledge or complete lack of knowledge of the actual relationship between secret key and physical property, the side-channel attack must be reasonably adapted. As leakage model it has to be used an intermediate value of the implemented cryptographic algorithm instead.

The aim of the thesis is to research the best leakage models that allow to execute a successful side-channel attacks on such devices. As shown in literature, during this investigation it was observed that most of the attacks are successful when a leakage model justifies the observed physical properties.

However, it has also been observed that it could have been performed successful attacks even when the leakage model was the furthest possible from the correct one, allowing us to study how cryptographic implementations are vulnerable when attacked with such models.

Keywords: MIA, mutual information, side-channel attacks, S-Box resilient to side-channel attacks

Sommario

Al giorno d'oggi siamo sempre più invasi da sistemi embedded, sensori e dispositivi integrati. Questi dispositivi operano in ambienti ostili che potrebbero intaccare il loro corretto funzionamento. La crittografia è uno degli strumenti che possono essere utilizzati per salvaguardare le funzionalità di tali dispositivi.

Però, le implementazioni crittografiche su sistemi fisici sono soggetti ad attacchi *side-channel*, attacchi che sfruttano grandezze fisiche (del dispositivo stesso) osservate durante una operazione di cifratura/decifratura. Gli attacchi migliori che possono essere applicati definiscono un *modello di leakage* come la relazione più accurata possibile che lega la chiave segreta applicata e la grandezza fisica presa in considerazione.

Quando il modello di leakage non è definibile per via di una conoscenza parziale o nulla della relazione tra chiave e grandezza fisica, l'attacco side-channel deve essere ragionevolmente adattato. Come modello di leakage dovrà essere utilizzato un valore intermedio estratto dall'algoritmo crittografico implementato.

Lo scopo della tesi è quello di ricercare i modelli migliori che permettono di compiere con successo attacchi side-channel su tali dispositivi. In questa investigazione di modelli è stato osservato che, come da teoria, la maggior parte degli attacchi ha successo quando viene definito un modello che giustifica le proprietà fisiche osservate.

Tuttavia, sono stati osservati anche attacchi altrattanto soddisfacenti quando il modello di leakage si allontana il più possibile dal modello corretto, dando così la possibilità di investigare la vulnerabilità di implementazioni crittografiche per tali modelli.

Parole chiave: MIA, mutua informazione, attacchi a canale laterale, S-Box resistenti ad attacchi a canale laterale

Contents

Abstract **i**

Sommario **iii**

Contents **v**

1 Introduction **1**
1.1 Basic Terminology and Concepts 1
1.1.1 Cryptographic Paradigms 2
1.1.2 Block Ciphers 2
1.2 Background on Side-Channel Attacks 8
1.2.1 Power-Based Information Leakage 9
1.2.2 Side-Channel Attacks 11
1.3 Open Challenges and Goals 13

2 Side-Channel Techniques **17**
2.1 Simple Power Analysis 17
2.2 Differential Power Analysis 21
2.3 Correlation Power Analysis 25
2.4 Mutual Information Analysis 29
2.4.1 Properties of the Leakage Model 35
2.4.2 Probability Distribution Estimation 37
2.4.3 Optimizations 39
2.5 Comparison between CPA and MIA 40

3 Leakage Model Exploration **45**
3.1 Leakage Model Definition 45
3.2 Side-Channel Attacks Implementations 47
3.2.1 Mutual Information Analysis Implementation 48

3.2.2 Correlation Power Analysis Implementation 52
3.3 Side-Channel Measurements Simulation 53
3.4 Data Visualization 58

4 Experimental Evaluation **59**
4.1 AES Substitution Box 59
4.2 DES Substitution Box 63
4.3 4x4 Substitution Box 65
4.3.1 CPA Resilient Substitution Boxes 68
4.3.2 TA Resilient Substitution Boxes 70
4.3.3 General Evaluation 71
4.4 5x5 Substitution Box 72
4.4.1 CPA Resilient Substitution Boxes 74
4.4.2 TA Resilient Substitution Boxes 78
4.4.3 General Evaluation 81

5 Conclusions and Future Developments **83**

Bibliography **85**

List of Figures **89**

List of Tables **91**

Ringraziamenti **93**

1 | Introduction

This chapter introduces the reader to the basic concepts of cryptography, drawing a general context of the work. First, it will be described the design of modern ciphers and how they work; then it will be explained what side channel attacks are and how they are used in practice. Finally, the reader will be talked into the open challenges and goals of the work.

1.1. Basic Terminology and Concepts

It is defined as space of message $\mathbb{M}$ the set of all possible messages over an alphabet $\mathbb{A}$. In turn, an alphabet $\mathbb{A}$ is defined as a finite set of symbols; in cryptography, the most common choice for the alphabet $\mathbb{A}$ is the binary alphabet ($\mathbb{A} = \{0, 1\}$). The choice of $\mathbb{A}$ actually reflects the binary representation of words in any computing device, such as PCs, smartphones and so on.

An element of the space $\mathbb{M}$ is a plaintext, and it will be referred to as m. The ciphertext space $\mathbb{C}$ is a set of strings over an alphabet $\mathbb{B}$ (this alphabet can coincide with the alphabet of $\mathbb{M}$, or it can be a whole different one). An element of $\mathbb{C}$ is referred to as ciphertext c.

A key space $\mathbb{K}$ is a set of elements called keys; the cardinality of this space is one of the figures of merit used to access the security margin of a cryptosystem.

Given an element $e \in \mathbb{K}$, it is defined as encryption algorithm the function $\mathbb{E}_e : \mathbb{M} \mapsto \mathbb{C}$ that maps bijectively an element from $\mathbb{M}$ to $\mathbb{C}$; on the other hand, for the inverse function, given an element $d \in \mathbb{K}$, it is defined as decryption algorithm the function $\mathbb{D}_d : \mathbb{C} \mapsto \mathbb{M}$ that maps bijectively an element from $\mathbb{C}$ to $\mathbb{M}$.

Given the above definitions, a cryptosystem can be defined as the 6-tuple:

$$\langle \mathbb{A}, \mathbb{M}, \mathbb{C}, \mathbb{K}, \{\mathbb{E}_e : e \in \mathbb{K}\}, \{\mathbb{D}_d : d \in \mathbb{K}\} \rangle \tag{1.1}$$

One of the most fundamental properties of a cryptosystem is *Correctness*: that is, the correct plaintext m of a ciphertext c can be obtained from c only if it is employed the

correct key k for decryption. A mathematical definition of it is:

$$\forall e \in \mathbb{K} \ \exists! \ d \in \mathbb{K} \ s.t. \ \forall m \in \mathbb{M} \ \mathbb{D}_d(\mathbb{E}_e(m)) = m \tag{1.2}$$

1.1.1. Cryptographic Paradigms

With respect to the cryptographic paradigm, it can be given an ulterior and more specific definition of a cryptoscheme. In this work, it will be analyzed *symmetric cryptosystems* or secret key cryptosystems.

Symmetric Cryptosystems

These cryptosystems are mainly used for the encryption of data at rest and data communications. One of the main advantages of these cryptosystems is that they are characterized by high computational efficiency. Data transmission between users is performed by the encryption of plaintexts under the same secret key k. Users that want to acknowledge the original message m from a ciphertext c must hold the same secret key k used for the encryption, that must not be disclosed under no circumstances. Its disclosure could affect past communications if it happened that the same key was employed in other communications. Since both encrypting and decrypting keys are equal, that is $e = d$, by rewriting the formal definition of a cryptosystem we will have that a symmetric cryptosystem is defined as:

$$\langle \mathbb{A}, \mathbb{M}, \mathbb{C}, \mathbb{K}, \{\mathbb{E}_e : e \in \mathbb{K}\}, \{\mathbb{D}_e : e \in \mathbb{K}\} \rangle \tag{1.3}$$

Asymmetric Cryptosystems

Also known as *public key cryptosystems*, they are mainly used between two users to establish and exchange a *symmetric secret key* for data communications. In this cryptosystems each user holds two keys: one of the two keys is public to everyone, while the other one is private and kept secret. Data encryption is performed by using the public key of the recipient of the message; while data decryption is performed on ciphertexts by applying the owned private key.

1.1.2. Block Ciphers

With respect on how both encryption and decryption algorithms are defined in a symmetric key system, there exist two kind of ciphers: block ciphers and stream ciphers. The main idea of stream ciphers is that they can be applied to messages of any length. That's

due to the fact that they have a 'state' and they can adapt to any plaintext length and produce a ciphertext with the same length.

Since ciphers analyzed in this work are exclusively block ciphers, it will be delved a little bit more into their detail.

The basic idea of block ciphers is that they operate on blocks of plaintext $m \in M$ ($m = \langle m_1, m_2, ..., m_n \rangle$, with $m_i = \{0, 1\}$) and produce blocks of ciphertext $c \in C$ ($c = \langle c_1, c_2, ..., c_n \rangle$, with $c_i = \{0, 1\}$) through a key-parametric transformation $\mathbb{E}_k(\cdot)$ or $\mathbb{D}_k(\cdot)$:

$$c = \mathbb{E}_k(m),\ m = \mathbb{D}_k(c) \tag{1.4}$$

The block size n is usually in the [64,256] bit range; if it happens that the plaintext is not multiple of a block size, it will be padded before the encryption. When a padded message is decrypted, the algorithm will recognize the padding scheme and discard it from the original message. For messages longer than a block size, a *mode of operation* is defined to describe the encryption or decryption sequence. The simplest method will divide the plaintext in q pieces of length n (equal to the block size), and encrypt them directly with the same key k. More complex methods will chain the encryption routine by introducing part of an encrypted block to the encryption of the following one. Then, the encryption of a block depends on its value and the block right before it. In this work it will be used plaintexts that will perfectly fit the length of the block cipher under study.

The main components of a block cipher are:

- *Cipher State*: it is defined as the result of each cryptographic operation performed by a cipher. It is initially initialized with the plaintext value, and then it will contain the value of the ciphertext at the end of the encryption;
- *Round*: it is a sequence of operations applied to the cipher state. It involves part of the secret key k and both operations that perform confusion and diffusion. The number of rounds depends on the key length and the overall complexity that it is wanted to achieve, and therefore the level of security of the block cipher;
- *Key Schedule*: It is the procedure that expands the original secret key k into *key material* to be used for each round. Key schedule is important because it will introduce to each round fresh key-dependent values.

Substitution Permutation Networks

One of the most employed schemes in the design of ciphers is the *Substitution Permutation Network*. A round of a SPN is split in three parts that act on the whole cipher state:

- *Key Mixing*: It introduces the key (or part of the key material) into the cipher state. It is usually performed by XOR operation;
- *Substitution*: It applies a non-linear function to the cipher state. It provides *confusion* to the cipher (with the help of the secret's key addition), and it is usually implemented as a look-up table. Confusion is that property that makes the relation between plaintext, key and ciphertext as complex as possible. Each digit of the secret key should influence the correspondence between plaintext and ciphertext in a non predictable way;
- *Permutation*: It performs a permutation of bits in the cipher state. It provides *diffusion* to the cipher, and it is usually implemented by bit-wise permutations or both XOR and rotate operations. Diffusion is that property that flattens as much as possible the frequency distribution of groups of plaintext letters. Having fixed value for the secret key, a single change of a plaintext bit should drive bits change in the ciphertext.

Advanced Encryption Standard It is one of the most used ciphers for data encryption based on Substitution-Permutation Networks. It is a 128-bits block cipher, which means that it operates on blocks of length 128-bits; it supports three different key sizes, that are 128-bits, 192-bits and 256-bits long.

The cipher can be seen as a set of operations applied at byte level. In fact, the cipher state of AES is defined as 4x4-byte matrix: this means that the block in input is divided into 16 blocks, each large 8-bits in size.

The encrypting operations performed in one round are:

- *AddRoundKey*: It adds the round key to the cipher state via the logical XOR operation;
- *SubBytes*: It performs the substitution operation. It is a nonlinear bijective function, implemented as a 8-to-8 bit map. It is usually referred to as S-Box. It takes a cipher state byte (usually updated already with the round key), and it substitutes it with another byte in a nonlinear fashion. It is implemented as two separate lookup tables, one for each encryption and decryption computations;

- *ShiftRows*: It is the first half of the permutation layer; it performs a cyclic shift on the state matrix. Each row of the state matrix is shifted by different offsets. It ensures that columns of the state matrix interact with each other;
- *MixColumns*: It is the second half of the permutation layer; it ensures that rows of the state matrix interact with each other. It is performed for each column of the state matrix by mixing with each other the bytes in the same column. It is implemented by multiplying each column state with a constant matrix.

Then, an AES round is a cascade of SubBytes, ShiftRows, MixColumns and AddRoundKey operations. The number of rounds of the cipher depends on the key size: with key sizes of 128-bits, 192-bits and 256-bits are employed 10, 12 and 14 rounds, respectively. The key schedule will produce as many round keys as the number of rounds. The first operation performed by the cipher involves a key addition operation with the former value of the secret key. Then follows a cascade or r rounds that involve the corresponding round key. In the final round, it is not performed the last MixColumns operation because easily invertible.

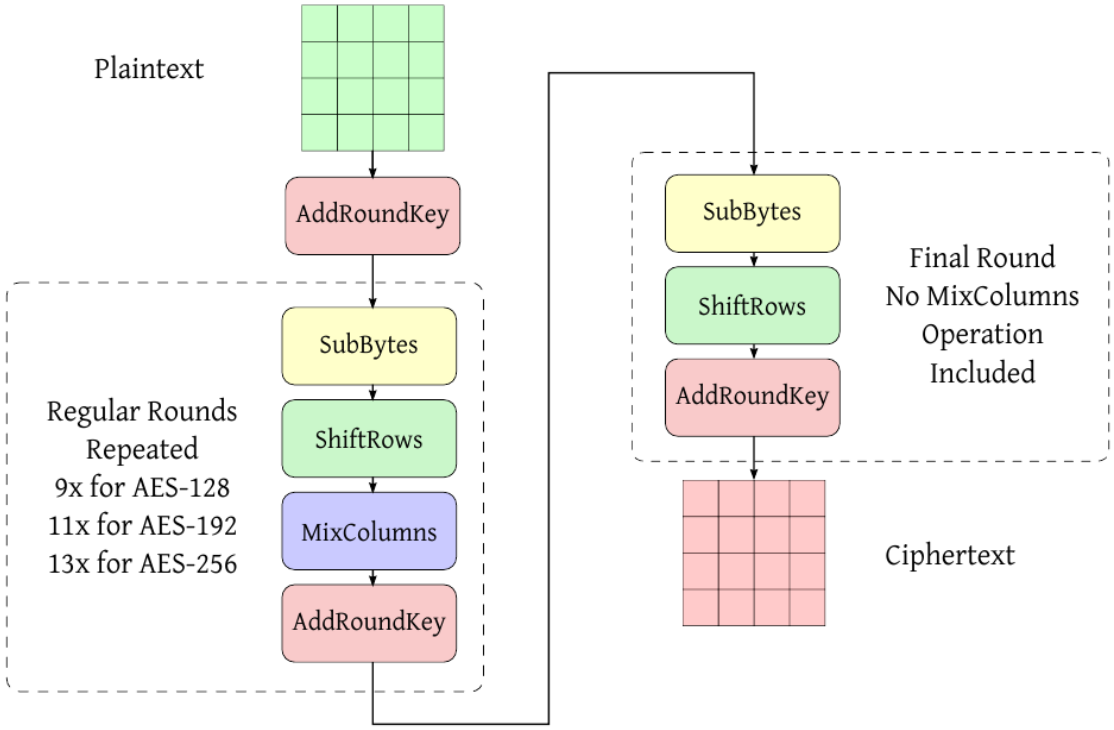


Figure 1.1: AES encryption structure

For what concerns the decryption, it is applied the inverse transformation of each cipher state in reverse order. That is:

- *AddRoundKey*: it adds to the ciphertext the relative round key via XOR operation. This step reverts back the former application of the round key performed during the encryption;
- *InvSubBytes*: it applies the inverse nonlinear transformation of the S-Box function;
- *InvShiftRows*: it rotates the cipher state to the opposite direction (with respect to ShiftRows);
- *InvMixColumns*: it inverts the column mixing operation. It is performed by multiplying each column state with the inverse matrix used in MixColumns.

The key schedule is performed in the same way for the encryption. The only difference is that the round keys are applied in reverse, that is, from the last one up to the former value.

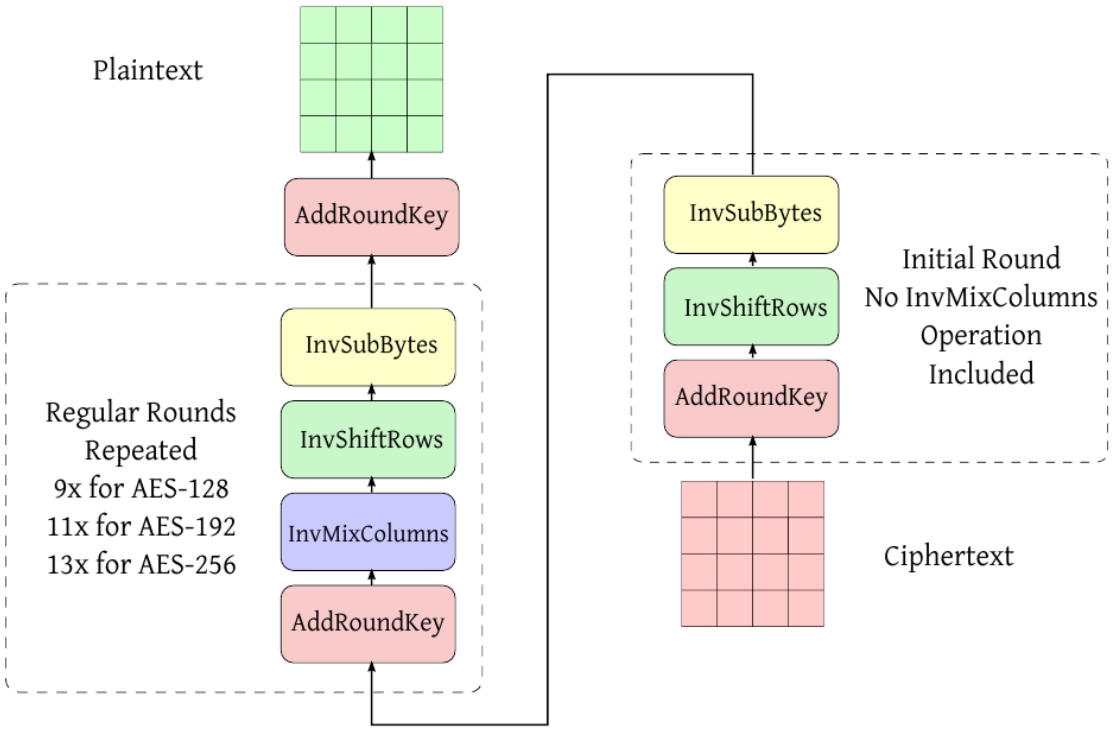


Figure 1.2: AES decryption structure

Feistel Network

Feistel Networks are another important designs used in cryptography. The cipher state consists in splitting the input block in two sub blocks of equal size, usually referred to as the couple (L,R). A round of a Feistel Network affects only one of these two sub blocks

by the application of key-dependent (non-linear) function. After each round it is then swapped the position of the two sub-blocks (and then the other sub block is taken into account). The decrypting operation exploits the same encrypting scheme, but it applies the round keys in reverse order. The reason why it is possible to reuse the same encrypting scheme is due to the invertibility of the round function. A typical round can be defined as:

$$L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus \mathcal{F}(K_i, R_{i-1}) \tag{1.5}$$

After a round application, the sub-block L at round i is updated with the value of the right sub-block R at round *i-1*; then the sub-block R at round i is updated by the computation of the non-linear function $\mathcal{F}$ (fed with round key) on sub-block R at round *i-1*; then the result is XORred with the sub-block L at round *i-1*.

Data Encryption Standard It is one of the most known algorithms designed with Feistel Networks. It is a 64-bits block cipher and supports only keys of 64-bits (of which only 56 are used). The number of rounds is 16; each round performs the same operations, but on different sub blocks. In addition to the rounds application of the Feistel Network, it is also performed a bit permutation at both start and end of the encrypting/decrypting computations. After having divided the 64-bit block as (L_{i-1}, R_{i-1}), the computation of $\mathcal{F}$ is performed by the following operations:

- *Expansion Permutation*: it takes as input the 32-bits long R_{i-1} block and outputs a 48-bits block via a fixed scheme (it duplicates some bits and permutes them);
- *Round Key Addition*: the round key is added via XOR operation to the sub-block obtained from the expansion permutation operation;
- *Splitting*: after the round key addition, the 48-bits block is split into 8 6-bits sub-blocks;
- *Substitution*: each 6-bits block value is then substituted by an S-Box: for each 6-bits block it is used an unique and independent S-Box. The output of an S-Box is a 4-bits value, and it is the only step that introduces non-linearity in the system. As for AES, they are implemented as look-up tables;
- *Permutation*: it combines eight 4-bits blocks into a 32-bits block and permutes its value.

The S-Box is a non-bijective function (more specifically, it is non-injective): as the implementation of the S-Box is a matrix of four rows and sixteen columns, the 1st and 6th bits in input to the S-Box are used to select the row of the matrix. The value made from the

remaining four bits will be substituted by the non linear scheme of the selected row (of the selected S-Box). The key schedule will produce sixteen 48-bits long round keys from the 56-bits one. Starting from the 64-bits key, it is first permuted and all the parity bits (that is, bits at position 8,16,24,..., 64) are removed from it. This produces a key long 56-bits. Then it is halved in half (into two sub keys of 28-bits), it is applied a cyclic shift (depending on the round) and it then recombined together (into a 48-bits key). Repeating these three operations sixteen times will give the sixteen round keys.

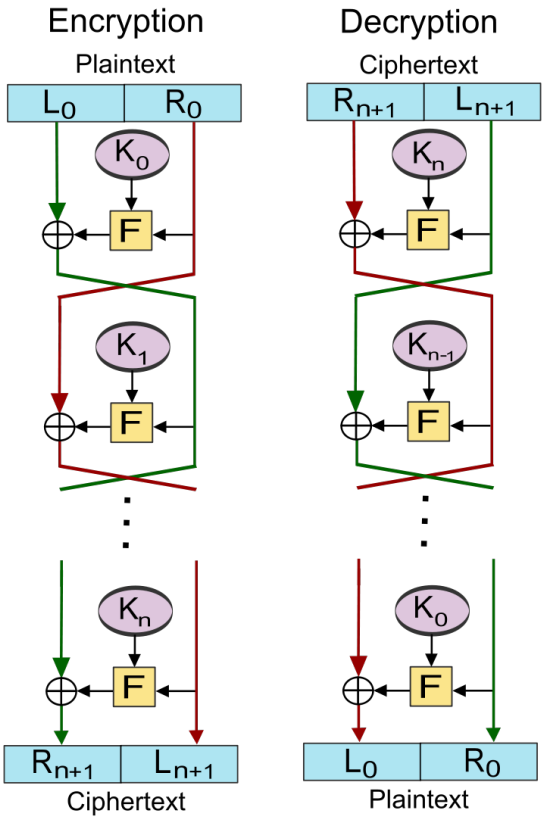


Figure 1.3: DES encryption and decryption structures

1.2. Background on Side-Channel Attacks

Most of the symmetric block ciphers that are employed nowadays are implemented in such a way that they are quite resilient to *linear* and *differential cryptanalysis*. Cryptanalysis is an attack on block ciphers that aims at recovering the secret key with an effort smaller than performing a *brute force* attack. Brute force is an attack that blindly tests all the possible keys of a ciphers on ciphertexts in the attempt to guess the correct secret key. If the key space $\mathcal{K}$ is too large, the computational effort that the attacker is required to

perform is so unbearable that it could not be performed within reasonable time. If the key space of such ciphers is reasonably big, it is said that the cipher is *computationally secure*.

With linear cryptanalysis, the attacker tries to approximate the non-linear behaviour of non-linear components with linear relations; then, by means of probabilistic analysis over these linear relations, the attacker might extract enough information over the secret key. Differential cryptanalysis, on the other hand, takes ciphertext pairs where the relative plaintexts show a particular difference: the XOR operations between those two ciphertexts is called *differential*, and certain differentials may have more probabilities than others of being observed (with respect to the chosen key). By analysing those probabilities, an attacker hopes to retrieve the key structure of the correct secret key employed for those ciphertexts.

Due to the fact that most ciphers are resilient to such techniques, attackers decided to target the physical implementations of such ciphers. A physical implementation of a cryptographic algorithm is bound to the physics of the device it is realized on: in an electrical system there is a strong relation between its power consumption and the relative cryptographic computation. A physical property that can be captured for such attacks is called *side-channel*, and the whole technique exploiting these properties for the extraction of a secret key of a cryptographic algorithm falls under the name of *Side-Channel Attack*. It is also referred as *leakage* the unintentional leak of information (whether it is physical or not) from a side-channel.

1.2.1. Power-Based Information Leakage

One of the physical measurements that is largely exploited in the vast majority of SCAs is the power consumption of the device. In an electrical device, data is represented in binary format by means of strings of ones and zeros; such strings are actually represented as vectors of *bits* (namely *bi*nary digi*ts*); each bit is physically made of transistors. A transistor is realized in Complementary Metal Oxide Semiconductor (CMOS) technology, and can mainly switch between the high logical level '1' and the low logical level '0' whether a current is flowing it. The amount of power drawn by a device during the computation of a cryptographic algorithm is linked to the data that is being processed. Thus, the power consumption measurements hold information about the various cryptographic operations that are performed. The total of power consumption of a CMOS circuit is the sum between

dynamic power and static power [26]:

$$P_{total} = P_{dynamic} + P_{static} \tag{1.6}$$

The static power is the power consumed when the device is not switching states. The dynamic power consumption occurs when the circuit switches between logical values. This latter contribution, despite being present only in state transitions, it is the predominant factor for the power consumption [11]. The dynamic power consumption can be defined as:

$$P_{dynamic} = C_L V_{DD}^2 P_{0\to1} f \tag{1.7}$$

Where C_L is the gate load capacitance, V_{DD} is the supply voltage, $P_{0\to1}$ is the probability of a switching activity and f is the clock frequency. Then, the current consumed by a device is linked to the energy needed to change bits from one state to another, which means that it is linked to the number of bits equal to the logical value '1' stored in that moment. The power consumption can be modeled by the *Hamming Weight* (or HW) and it is simply based on the amount of '1's in a binary value. E.g., the Hamming Weight of the binary string '0b0101' (5 in decimal) is equal to 2. The Hamming Weight can be used to model the immediate power consumption of a device in a time instant t.

In addition to it, for power consumption it can be defined a much more detailed model, that is the *Hamming Distance* (or HD) model [4]. It is defined as Hamming Distance between two strings as the minimum number of bit flips that can be performed on one value to obtain the second one. The Hamming Weight can be seen as the Hamming Distance with respect to the null string (a string full of zeroes). E.g., the Hamming Distance between '0b1010' and '0b0100' is equal to 3 (as the first 3 bits are different in both strings). The Hamming Distance can be used as a more accurate model for the power consumption of data computed in two different time instants t and *t-1*:

$$W = a\, HW(D \oplus R) + b = a\, HD(D, R) + b \tag{1.8}$$

$$W = a\, HW(D) + b \tag{1.9}$$

Where the first formula defines the power consumption W related to the Hamming Distance, and the second formula defines the power consumption W related to the Hamming Weight. The parameter a is a scalar gain between Hamming Distance HD and power consumed; the parameter b is the power dissipation induced by noise. D is the handled data by the device and R is the data reference from which the state switches from (and that is equal to a vector full of zeroes in case of the Hamming Weight).

To measure a circuit's power consumption, it is put a small resistance (like 50Ω) in series to the power or ground input; then two probes connected to an oscilloscope are attached to both ends of the resistor and voltages changes are captured over time. The more accurate is the oscilloscope, the better the quality of the measurements and thus the better the quality of the attack. Also the quality of the both probes and resistance may affect the quality of the measurements, as well as temperature, pressure, and other environmental factors.

A single power measurement of a cryptographic algorithm is usually referred to as *power trace*, and it is represented by the vector $\mathbf{t}^n$. A trace is represented as:

$$\mathbf{t}^n = [t_0^n, t_1^n, ..., t_l^n]$$

Where the superscript identifies the n-th acquisition of the power trace, that is, the acquisition related to the whole cryptographic operation; while the subscript identifies the l-th sample taken from that cryptographic operation.

A power trace is also susceptible to noise: noise is actually modelled as a random variable ε having normal distribution with 0 mean and σ deviation standard $\mathcal{N}(0, \sigma)$. Such definition of the noise is called *Gaussian*.

Then, a voltage measurement at both ends of a resistor can be defined as[24]:

$$t(x) = Voltage_{actual}(x) + \varepsilon \tag{1.10}$$

Where $Voltage_{actual}$ is the noise-free value of the voltage, and ε is the additional noise. Then, a typical operation performed over traces is to average a good amount of measurements (related to the same encryptions with the same input data). In this way it can be filtered some part of the noise and deal with much cleaner traces.

1.2.2. Side-Channel Attacks

Differential Power Analysis is the first side-channel attack that made use of a *leakage model* for a successful attack[24, 14]. A leakage model defines the relationship between the power consumption of the device and the secret key that is being employed. Such leakage model is used to compute leakage predictions that will be compared with real side-channel observations by means of a *Distinguisher*. A distinguisher is any statistic used to compare side-channel measurements with hypothesis dependent predictions, and that are able to point out the correct hypothesis[24]. For each key hypothesis it will be assigned a score by the distinguisher that will describe how the leakage prediction

and the side-channel measurement are dependent. If an attack is successful, the score assigned to the correct key hypothesis will be the highest with respect to all the other scores. Research has investigated several distinguishers throughout the years, such as *Difference of Means*[14], *Pearson's correlation coefficient*[4], *Bayesian Classification*[5], *Mutual Information*[10], and so on. The implementation of a good distinguisher allows an attacker to retrieve the correct key hypothesis with the highest *Success Rate* possible. The success rate is a metric used in SCAs that gives information about the chances a distinguisher has to extract the correct key, and it is computed empirically by performing an attack a certain number of times and to record the average number of successes[28, 17].

In a DPA it is chosen as leakage model one bit from an intermediate value of the cryptographic algorithm. This intermediate value involves the application of both known data and key hypothesis, and such model it used to point out the power consumption driven by that bit. Then, for each key hypothesis, the predictions computed by such leakage model are used to classify side-channel measurements into two distinct classes that will be compared by means of a distinguisher. In this case, in a simple DPA, as distinguisher it is used the Difference of Means between those classes. Under the assumption of a correct key hypothesis, the set of side-channel measurements will be correctly classified into the relative classes, and the distinguisher will infer the correct by assigning to it the highest score.

An enhancement of such attack is *Correlation Power Analysis*[4]. CPA takes DPA a step further by modelling in the most accurate way possible the power consumption of multiple bits of an intermediate value of a cryptographic algorithm. Due to the physical properties of electrical devices used for cryptographic implementations, the most accurate leakage model that can be used is the Hamming Distance. In fact, an electrical system leaks the Hamming Distance (or Hamming Weight if the reference state is zero) of binary values. because of the more complex definition of the leakage model, as distinguisher it is used a much more sophisticated tool, namely Pearson's correlation coefficient. This tool is able to identify any linear relation between two random variables, which in this case are side-channel measurements and leakage predictions.

In fact, when the device leakage model is known, CPA is renowned to be the best attack that allows to extract the secret key with the highest success rate possible.

1.3. Open Challenges and Goals

As it was described in the previous chapter, when the leakage model of a target device is well understood, CPA is the best side-channel attack that can be performed on such cryptographic implementations. When the knowledge of a leakage model is partial, whether it be partially known or utterly unknown, CPA's performance will drastically drop[7].

In such scenarios, side-channel attacks must adapt the description of both leakage model and distinguisher in order to execute a successful attack.

In such circumstances, assumptions on leakage model need to relax, and as leakage model it is chosen an intermediate value computed by the cryptographic algorithm itself. The hypothetical leakage model is usually defined as $\hat{L}_{\hat{k}} := f(\hat{k}, M)$, where $\hat{k}$ is a key hypothesis, M is a known datum, and $f(\cdot)$ is a suitable function computed from the cryptographic algorithm.

In contrast with (correlation) power analysis attacks, the complexity of the attack is in the choice of a suitable distinguisher that must be capable to infer more information from a relaxed model. In fact, one of the most fitting distinguishing tool is the *Mutual Information*, an instrument capable of extracting *any* dependency between two random variables. When two random variables are highly dependent, the value of the mutual information is very high; when there is almost no relationship between two random variables, the mutual information value is very small.

Such distinguisher is used in *Mutual Information Analysis*[10] (or MIA), a generic side-channel attack characterized by having a high success rate when performed on partially known or unknown models, and when the hypothesis of noise are not Gaussian. Key extraction is achieved by computing the maximum of mutual information $\mathbf{I}(\cdot)$ between $\mathbf{O}$ and $\hat{\mathbf{L}}_{\hat{k}}$:

$$max_{\hat{k}}\, \mathbf{I}(\mathbf{O}|\hat{\mathbf{L}}_{\hat{k}}) \tag{1.11}$$

Where $\mathbf{O}$ is the random variable representing all the possible values that could be taken by the measurements; $\hat{\mathbf{L}}_{\hat{k}}$ is the random variable representing all the possible values that could be taken from the hypothetical leakage model under that key hypothesis $\hat{k}$.

Since leakage models cannot be molded as accurately as possible after the physics of the targeted device, the choice for a suitable model used to justify side-channel observations is still a burning issue.

In the *State of Art* of MIA[10] it was presented an instance of a generic side-channel attack

that defined the leakage model after a truncated version of an intermediate cryptographic value; then it extracted the correct secret key by maximizing the mutual information between leakage predictions and side-channel observations. The truncation of that value was necessary for a correct evaluation of the mutual information, which would have been invalid otherwise.

The challenge that this work takes upon is to investigate the most suitable leakage models defined as $\hat{L}_{\hat{k}} := mask_{out} \,\&\, f((\hat{k}, M) \,\&\, mask_{in})$ that can be used for attacks with the highest success rate. The investigation is accomplished by means of exhaustive search of combinations of both variables $mask_{in}$ and $mask_{out}$. The variable $mask_{in}$ allows to control subsets of arguments that are fed to the intermediate function $f(\cdot)$; in turn it enables to explore several subsets of observable values of $f(\cdot)$ with respect to key hypothesis $\hat{k}$. While the variable $mask_{out}$ allows to investigate different models by truncating combinations of bits of $f(\cdot)$.

In literature there exist works that investigated and designed functions $f(\cdot)$ that, when used for modelling the leakage, degrade the overall success rate of CPA attacks[16].

Since there is no work on literature about the study and design of functions $f(\cdot)$ resilient to MIA, in this work it was seized the opportunity to analyze such functions $f(\cdot)$ presented in [16] and search out the ones most resilient to MIA.

However, not always it is possible to perform a successful attack when observing the maximum of mutual information. Unsuccessful attacks may happen when leakage models used to represent side-channel observations might be completely independent from keys assumptions, or might not be particularly accurate under the assumption of correct key hypothesis.

An interesting question would be: What if it can be derived a model that is *systematically* wrong? A systematic wrong model would mean that the model will be the most wrong possible only when it is tested with the correct key hypothesis. Such model will be the worst at justifying every side-channel observation, and it will perform on average when interrogated with wrong key hypothesis.

Then, the secret key extraction would fail when key hypothesis are distinguished with the maximum of mutual information. In contrast, when the worst leakage models are employed, successful attacks would be observed when looking at the minimum of mutual information instead.

In literature there is no related work that investigates generic side-channel attacks with worst leakage models.

The definition of the model $\hat{L}_{\hat{k}}$ does also allow to investigate for models that are the most distant possible from the real one only under the assumption of correct key hypothesis. This investigation is performed by looking at the minimum of mutual information when it is used such models.

The contributions of this thesis are:

- Offer a systematic analysis of the best leakage models possible $\hat{L}_{\hat{k}} := mask_{out} \,\&\, f((\hat{k}, M) \,\&\, mask_{in})$ under the assumption of correct key hypothesis $\hat{k}$ on several functions $f(\cdot)$. Such analysis are performed by looking at the maximum of mutual information $max_{\hat{k}}\, \mathbf{I}(\mathbf{O}|\hat{\mathbf{L}}_{\hat{k}})$, and it is investigated for those functions $f(\cdot)$ that offer a particular resilience when attacked with MIA;
- Offer a completely new systematic analysis of the worst leakage models possible $\hat{L}_{\hat{k}} := mask_{out} \,\&\, f((\hat{k}, M) \,\&\, mask_{in})$ under the assumption of correct key hypothesis $\hat{k}$ when looking at the minimum of mutual information $min_{\hat{k}}\, \mathbf{I}(\mathbf{O}|\hat{\mathbf{L}}_{\hat{k}})$. This analysis is performed on those functions $f(\cdot)$ that are able to withstand attacks when the secret key is extracted with the maximum of mutual information.

In the following chapters it is explained how such analysis have been carried out. In chapter 2, the reader will be introduced to the State of Art of side-channel attacks: it will be described both CPA and MIA implementations. It will be described what are the most suitable intermediate values used for model definitions and their main characteristics. In chapter 3, it will be described in detail how the experiments were organized and then executed. In chapter 4, it will be shown and discussed the results obtained from such experiments. Finally it will be wrapped up by expressing some final considerations on the whole work.

2 | Side-Channel Techniques

This chapter introduces the reader to the state of art of side-channel attacks. It will be explained how the most useful techniques are executed and the nuances that hide behind them. In general, there is one important characteristics that all side-channel attack share in common: every side-channel attack recollects the secret key used into a cryptographic encryption by means of a *divide-and-conquer* strategy. With this strategy, the secret key gets divided into several sub-keys, and the side-channel technique is then used to extract the value of each sub-key independently. In this way, an attacker would be able to test and extract each sub-key in a fast and efficient way. However, this strategy depends to a great extent on the capability of the attacker to define a suitable model that should be used to represent the target device under attack.

2.1. Simple Power Analysis

Introduced during 1999 by Kocher et al.[14], *Simple Power Analysis* (or SPA) is both the first and the simplest side-channel attack that targeted a cryptographic implementation. It is based on a graphical analysis of the power consumption over time of a cryptographic computation. The sole visible inspection of a power trace gives already enough information about the target device and the type of cryptographic algorithm that is being computed.

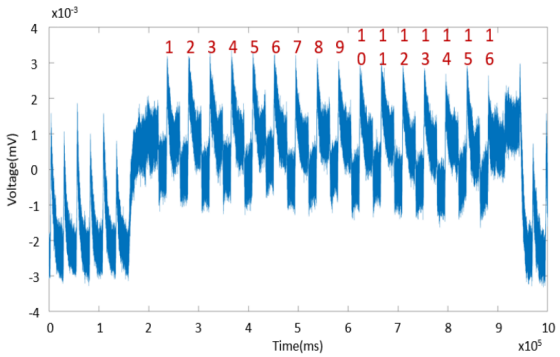


Figure 2.1: SPA on a DES encryption

As it can be seen in figure 2.1, it can be observed with naked eye all the 16 rounds of a DES encryption/decryption computation.

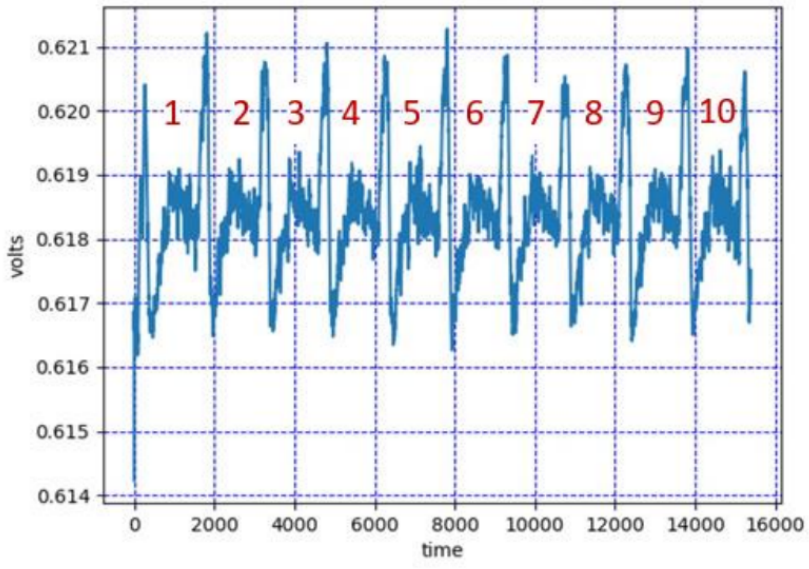


Figure 2.2: SPA on a AES encryption

While in figure 2.2 it can be noticed 10 rounds of an AES computation. Despite both figures were taken on different target devices, it was still possible to observe the relative electrical patterns of both cryptographic computations. With a closer inspection to the AES power trace in figure 2.3, it can be observed the patterns relative to the computation of the 16 S-Boxes substitutions (blended with the ShiftRows operation), MixColumns and addRoundKey.

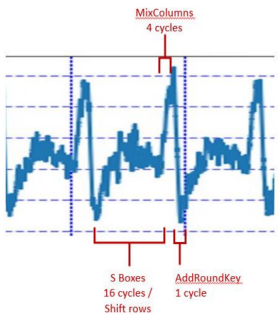


Figure 2.3: SPA on a round computation of AES' cipher

Thus, SPA can reveal the sequences of instructions performed by a cryptographic implementation. Moreover, this visual attack can be used to extract the key for those cryptographic techniques where the performed operations rely heavily on the particular

datum that it is being handled. That is, where the execution path depends on the processed data. In fact, in RSA (that is an asymmetric cryptosystem), the secret key x is used for the computation of a value R by the formula $R = y^x mod\, n$, where n is a public value, and y can be found by an attacker. The computation of R is performed by the *square and multiply strategy*: if in the secret key it is met the bit '0', it is only performed a square; if it is met the bit '1', it is also performed a multiply operation. It follows the pseudo-code of a square and multiply strategy:

Algorithm 1: Square and Multiply right to left

Data: $y, x, t = \lceil log_2 x \rceil, x = (x_{t-1}, ..., x_1, x_0)$
Result: $c = y^x$
if $x = 0$ **then**
    return 1
end
$b \leftarrow y$
if $x = 1$ **then**
    $c \leftarrow y$
else
    $c \leftarrow 1$
end
for $i \leftarrow 1$ ***to*** $t - 1$ **do**
    $b \leftarrow b^2\ mod\ n$
    if $x_i = 1$ **then**
        $c \leftarrow c \cdot b\ mod\ n$
    end
end
return c

Thus, it is possible to identify each bit of the secret key by inspecting the patterns in power consumption during the execution of R. In fact, in figure 2.4 it can be observed the different power consumption values with respect to each processed bit.

Since with SPA it was possible to observe patterns in the power consumption, the simplest countermeasure that could be taken was to execute additional operations in order to mask such patterns. In this way, it would not be possible to visually discern patterns within measurements and relate them to key bits. On the other hand, this countermeasure comes with an increase in power consumption and a degradation of performance[14] that can be quite expensive with respect to the operations that are duplicated.

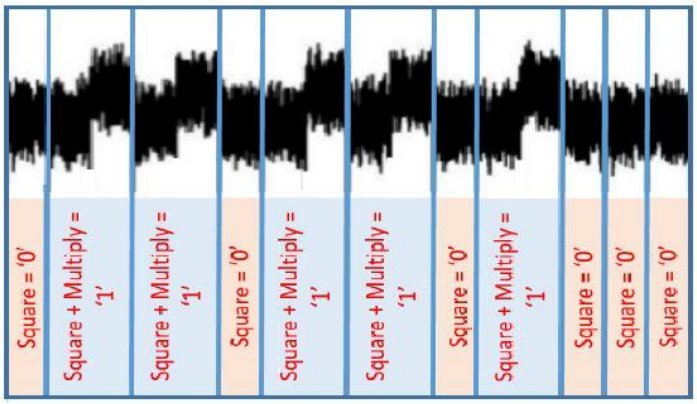


Figure 2.4: Portion of a power trace of a computation of a RSA private key

Algorithm 2: Square and Multiply right to left with the countermeasure

Data: $y, x, t = \lceil log_2 x \rceil, x = (x_{t-1}, ..., x_1, x_0)$

Result: $c = y^x$

if $x = 0$ **then**
    return 1

end

$b \leftarrow y$

$q \leftarrow 0$

if $x = 1$ **then**
    $c \leftarrow y$

else
    $c \leftarrow 1$

end

for $i \leftarrow 1$ ***to*** $t - 1$ **do**
    $b \leftarrow b^2 \ mod \ n$
    if $x_i = 1$ **then**
        $c \leftarrow c \cdot b \ mod \ n$
    else
        $q \leftarrow c \cdot b \ mod \ n$
    end

end

return c

But most of cryptographic algorithms do not encode the secret key into cryptographic operations. For example, symmetric ciphers, due to their design, blend the secret key with plaintext messages, and perform a fixed set of operations on them. The set of performed operations is the same, does not change with respect to key bits values. What changes

with respect to the key are the operands on which such operations are performed on.

2.2. Differential Power Analysis

It's the first statistical SCA developed during 1999 by Kocher P. et al.[14]. It was presented in combination with SPA at the conference of *CRYPTO 1999*, and it is considered as the basis of the following studies over side-channel attacks. With respect to a SPA, in addition to the measurements, this attack is also the first attack that exploited a *selection function* (also known as leakage model), used for the classification of side-channel measurements.

The attack is executed as follows[14]:

1. The attacker observes N encryptions/decryptions performed with the same secret key k^*, and captures the relative power traces $\mathbf{T}^n[0..L]$ along with the relative plaintexts/ciphertexts values $\mathbf{M}^n$:

$$\mathbf{T}^n[0..L] = \begin{bmatrix} \mathbf{t}^1 \\ \mathbf{t}^2 \\ \vdots \\ \mathbf{t}^N \end{bmatrix} = \begin{bmatrix} t_0^1 & t_1^1 & \cdots & t_L^1 \\ t_0^2 & t_1^2 & \cdots & t_L^2 \\ \vdots & \vdots & \ddots & \vdots \\ t_0^N & t_1^N & \cdots & t_L^N \end{bmatrix} \quad \mathbf{M}^n = \begin{bmatrix} m^1 \\ m^2 \\ \vdots \\ m^N \end{bmatrix}$$

2. Then it is chosen the selection function. Such function is used to classify side-channel measurements into two sets of classes. In simple DPA, the selection function computes one bit value taken from an intermediate value of a cryptographic operation that employs both a secret key and some known data. The model is defined differently for each cryptographic algorithm, based on how easy it is to test plaintext/ciphertext values and key hypothesis.

   In a cryptographic algorithm, encryption and decryption is performed by repeating several operation in multiple rounds. In general there are two rounds that an adversary can choose for a side-channel attack:

   - The first round of a cryptographic algorithm allows to exploit the plaintext message as known data;
   - The last round of a cryptographic algorithm allows to exploit the ciphertext as known data.

   Targeting deeper rounds would be possible, but it would require an attacker to

increase the complexity of the attack.

A common target value for the selection function is one bit from the output of a S-Box computation. In a block cipher, both secret key and known data can be divided into blocks of same length:

$$k^* = \{k_1^*, k_2^*, ..., k_r^*, ..., k_R^*\},\ 1 \leq r \leq R \tag{2.1}$$

$$m^n = \{m_1^n, m_2^n, ..., m_r^n, ..., m_R^n\},\ 1 \leq r \leq R \tag{2.2}$$

A S-Box is normally implemented as a look-up table indexed by the XOR between a subset of known data m_r^n and hypothesis on a subset of the secret key k_{hyp}: $f(m_r^n, k_{hyp}) = \ S\text{-}Box(m_r^n, k_{hyp})$.

It is formally defined as $D(M, b, k_{hyp})$ the selection function $D(\cdot)$ that takes as argument the parameters:

- M, that represents the value of the known data;
- k_{hyp}, that represents the value of the tested key hypothesis;
- b, that is the position of the bit that is used for traces' classification.

Then, the selection function can be defined as:

$$D(M, b, k_{hyp}) = S\text{-}Box(M, k_{hyp})[b] = \ S\text{-}Box(m_r^n, k_{hyp})[b] \tag{2.3}$$

3. It is computed the value of $D(M, b, k_{hyp})$ for each trace for a specific S-Box. It is first fixed the value of r. Then it is chosen a suitable position for the output bit b and it is fixed a key hypothesis k_{hyp}. Next, it is plugged into the leakage model the subset values of recorded plaintexts/ciphertexts m_r^n/c_r^n one at a time. The output of the selection function is binary by construction, and with respect to the outcome (0 or 1), power traces are collected into two different bins;
4. After having classified all the measurements, it is then performed an average between all traces within the same bin. That is, all the traces in the bin where $D(\cdot)$ was equal to '0' are averaged together; the same is performed with traces in the other bin (where $D(\cdot)$ was equal to '1'). These traces are referred to as master traces;
5. Finally it is performed a final L-sample differential trace $\Delta_D[0..L]$ by finding the difference between the two master traces. The differential trace is computed as[14]:

$$\Delta_D(j) = \frac{\sum_{n=1}^{N} D(m_r^n, b, k_{hyp}) * \mathbf{T}_j}{\sum_{n=1}^{N} D(m_r^n, b, k_{hyp})} - \frac{\sum_{n=1}^{N} (1 - D(m_r^n, b, k_{hyp})) * \mathbf{T}_j}{\sum_{n=1}^{N} (1 - D(m_r^n, b, k_{hyp}))} \approx$$
$$2 * \left(\frac{\sum_{n=1}^{N} D(m_r^n, b, k_{hyp}) * \mathbf{T}_j}{\sum_{n=1}^{N} D(m_r^n, b, k_{hyp})} - \frac{\sum_{n=1}^{N} \mathbf{T}_j}{N} \right), \text{ with } r, b, k_{hyp} \text{ fixed}$$

If in the vector $\Delta_D(j)$ it is found some significant values, then the key hypothesis k_{hyp} is correct. Otherwise, if $\Delta_D(j)$ is flat almost everywhere, the attack is performed again from step 3. with a different key hypothesis k_{hyp} and possibly by taking into account a different position for the bit b;

6. Cryptographic algorithms implement several S-Boxes, and for each S-Box it is used part of the secret key k^* by means of sub-keys k_r^*. A single run of a DPA allows to extract one sub-key k_r^* used for that particular S-Box that is being analyzed. Thus, the extraction of the entire key k^* is obtained by repeating the attack from point 3. by cycling through all the possible values that the variable r can take on.

If k_{hyp} is incorrect, the bit computed using $D(\cdot)$ will differ from the actual value for half of the C_i values. That is, when k_{hyp} is incorrect, there is probability $P \approx \frac{1}{2}$ of guessing a correct value for the chosen bit for each C_i. Thus the selection function is *uncorrelated* to what it is actually computed from the target device. If a random function is used to divide sets of measurements into two distinct subsets, the difference between the two master traces would approximate 0 as the number of measurements increases[14]:

$$\lim_{n \to \infty} \Delta_D(j) \approx 0 \tag{2.4}$$

However, if k_{hyp} is correct, the computed value of $D(\cdot)$ will be equal to the actual value with probability equal to 1. The selection function will be *correlated* to the manipulated bit. Then it will be observed the effect of the bit in the power consumption as the number of measurements $n \to \infty$. All the other data values, measurements, etc. not correlated to $D(\cdot)$ approach zero. The plot of $\Delta_D(\cdot)$ would be expected to be flat almost everywhere (where data is uncorrelated with $D(\cdot)$), and some spike in those regions where data being processed is correlated with $D(\cdot)$. In figure 2.5 it is shown a DPA attack on DES traces: the first trace represents the average value of a trace; the following three $\Delta_D(\cdot)$ traces related to the correct key hypothesis and two wrong keys hypothesis respectively. Thus, by performing DPA over all sub-keys, it is possible to recollect the entire secret key employed for the encryption.

The distinguisher presented in the original paper was the *DoM* (or Difference of Means), and it used to compare the first moment (or mean) between two classes; later on got

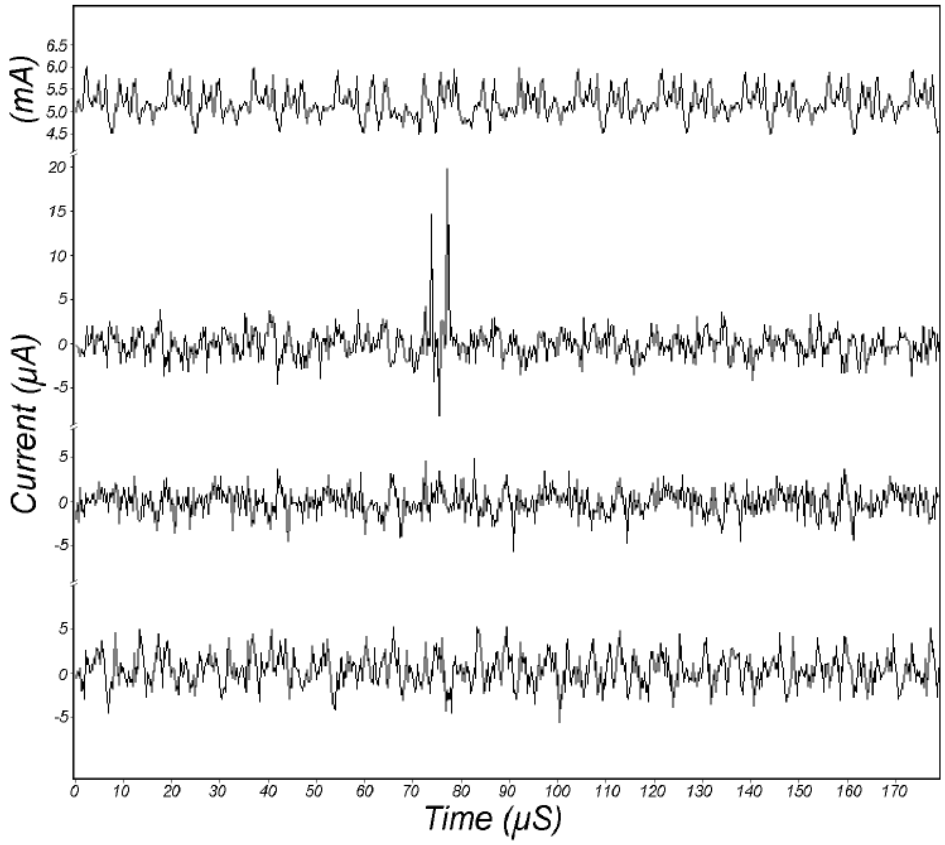


Figure 2.5: DPA attack, with one successful attack and two unsuccessful attack

developed other high-order DPA attacks, where more samples are used at the same time from within the same power [14, 20, 23]. One important characteristic of this attacks is that it does not require any particularly expensive set of instruments for a successful attack.

There were different countermeasures that were proposed in the same paper[14] to reduce the susceptibility of cryptographic implementations to this kind of attack.

The first approach was to reduce the signal size by reducing the overall *Signal-to-Noise* ratio (SNR). This was achieved by choosing operations that leaked less information in terms of power consumption, for e.g. by using constant execution path code. It could be also implemented a physical shield so as to isolate the device. This solutions would not reduce completely the signal size, but instead it would require an attacker to gather more measurements. In particular, physical shielding does also increase the overall cost and size of the device where it is implemented on[14].

Another approach would introduce artificial noise into power consumption measurements. Moreover, it could be also randomized both the execution time and order of some opera-

tions[14]. This could be implemented by inserting fake cycles, use unstable clocking and random delays. This would allow to desynchronize multiple power measurements taken from the same sample set.

These two approaches are also known as hiding, and what they do is decrease the correlation between hypothetical power consumption predicted by a model and the power consumption of the device where the cryptographic algorithm is implemented on[18]. Hiding does not affect the computation of the algorithm (or only in a minimum part), and decreases the overall SNR.

Hiding, however, can be defeated by averaging multiple measurements taken from the same cryptographic operation. This technique is called *windowing*[8], and an attacker is required to gather and average a lot of measurements.

A third approach would require to design a cryptosystem with some knowledge on the underlying hardware it will be built on. In this way the designer can use leakage functions to analyze and improve the overall design[14].

In the same conference of *CRYPTO 1999* it was also presented by Chari et al. a paper on sound approaches against power analysis attacks[6]. Here it was introduced the concept of *masking*: this technique would conceal each bit value by splitting it into n shares as: $b_0 \oplus r_1 \oplus r_2 \oplus ... \oplus r_{n-1} \oplus (r_1 \oplus r_2 \oplus ... \oplus r_{n-1})$, where each r_i bit is chosen randomly. From an attacker perspective, this toughens his ability to predict the right bit values.

But masking alone cannot prevent high-order DPAs. In a high-order DPA, multiple points from several traces are combined into a single value by means of a preprocessing function. Then, this value, can be easily attacked by a simple DPA attack. The points taken into account can be either a masked intermediate value and the value of the mask itself, or two intermediate values masked by the same mask.

Then, masking alone is not enough, and other diverse countermeasures must be implemented at the same time to obtain a good level of resilience[30].

Both masking and hiding can be implemented on software level and hardware level. The costs of hardware implementations do increase as it is chosen more fine grained techniques.

2.3. Correlation Power Analysis

Developed in 2004 by Brier et al., it took DPA a step further[4]. CPA is a model-based approach, but in opposition to (simple) DPA, it takes into account multiple bits from the selection function. In addition to it, this attack does also exploit the information of the

power consumption model of the target device. Thus, as leakage model it is taken the Hamming Distance of the whole selection function with respect to an unknown reference value R. Then, for the key extraction, such model is tested with several key hypothesis and reference values. Instead of using the difference of average traces, as distinguisher it is used the *Pearson's Correlation Coefficient*. This coefficient is a statistical test that measures the linear correlation between two random variables X and Y, and it is defined as:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sqrt{E[(X-\mu_X)^2]E[(Y-\mu_Y)^2]}} \tag{2.5}$$

Where E is the usual notation for the expected value, σ_X is the standard deviation of a random variable X, and $cov(\cdot)$ is the covariance between two random variables. In this case, the two random variables are the leakage model fed with key hypothesis and the power traces of cryptographic encryptions with unknown secret key k^*. The Pearson's correlation coefficient takes values between the range $-1 \leq \rho \leq +1$. When the two random variables are totally uncorrelated, ρ approximates to 0; when it is used a perfect leakage model, ρ approximates to ± 1. If the correlation is close to -1, it means that the random variables are anti-correlated: where one increases, the other decreases, and vice versa.

The attack goes as follows[4]:

1. The attacker observes N encryptions/decryptions performed with the same secret key k^* and captures both power traces $\mathbf{T}^n[0..L]$ and their relative plaintexts/ ciphertexts $\mathbf{M}^n$:

$$\mathbf{T}^n[0..L] = \begin{bmatrix} \mathbf{t}^1 \\ \mathbf{t}^2 \\ \vdots \\ \mathbf{t}^N \end{bmatrix} = \begin{bmatrix} t_0^1 & t_1^1 & \cdots & t_L^1 \\ t_0^2 & t_1^2 & \cdots & t_L^2 \\ \vdots & \vdots & \ddots & \vdots \\ t_0^N & t_1^N & \cdots & t_L^N \end{bmatrix} \quad \mathbf{M}^n = \begin{bmatrix} m^1 \\ m^2 \\ \vdots \\ m^N \end{bmatrix}$$

2. Then it is chosen the leakage model. As usual target of a SCA, it is targeted a function that employs part of the secret key and known data. In block ciphers, keys and known data are divided into blocks of equal size as in 2.1 and 2.2; the best candidate function is the Substitution Box, as it is implemented as a look-up table indexed by key hypothesis and known data $f(m_r^n, k_{hyp}) = LUT(m_t^n, k_{hyp})$. By taking into consideration how information is leaked from the power consumption of

the device, it can be defined the leakage model as:

$$L_{k_{hyp}} := h(m_r^n, k_{hyp}, R) = HW(S\text{-}Box(m_t^n, k_{hyp}) \oplus R) = \\ = HD(S\text{-}Box(m_r^n, k_{hyp}), R) \tag{2.6}$$

where R is an unknown but constant reference state. Sometimes the reference R is systematically set to 0, which in turn it would simplify the leakage model with just the Hamming Weight $L := h(m_t^n, k_{hyp}) = HW(S\text{-}Box(m_t^n, k_{hyp}))$. Otherwise, it is required to identify the correct value for R by exhaustive search. Generally, this operation is performed only once[4];

3. It is first fixed the index r related to the attacked $S\text{-}Box(\cdot)$, and a key hypothesis k_{hyp}. Then it is computed the Pearson's correlation coefficient for each sample point l. These computations allow to obtain a measure of the correlation for each sample point l between the chosen key hypothesis k_{hyp} and all power measurements t_i:

$$\rho_{k_{hyp},l} = \frac{\sum_{i=1}^{N}[(h_{i,k_{hyp}} - \mu_{h_{k_{hyp}}})(t_{i,l} - \mu_{t_l})]}{\sqrt{\sum_{i=1}^{N}(h_{i,k_{hyp}} - \mu_{h_{k_{hyp}}})^2 \sum_{i=1}^{N}(t_{i,l} - \mu_{t_l})^2}}, \\ 0 \leq k_{hyp} \leq K, \quad 0 \leq l \leq L$$

Where $\rho_{k_{hyp},l}$ is the correlation coefficient with respect to sub-key hypothesis k_{hyp} at sample position l;

$h_{i,k_{hyp}}$ is the value of the power prediction induced by the leakage model for each power measurement i under key hypothesis k_{hyp}. It is computed as $h_{i,k_{hyp}} = HW(S\text{-}Box(m^i, k_{hyp}) \oplus R)$;

$\mu_{h_{k_{hyp}}}$ is the mean value between all power predictions under key hypothesis k_{hyp}. It is computed as $\mu_{h_{k_{hyp}}} = \frac{1}{N}\sum_{i=1}^{N} h_{i,k_{hyp}}$;

$t_{i,l}$ is the value for each power measurement i at sample point l;

μ_{t_l} is the mean value between all power traces at sample point l. It is computed as $\mu_{t_l} = \frac{1}{N}\sum_{i=1}^{N} t_{i,l}$;

Then, the computation of $\rho_{k_{hyp},l}$ is executed again for each key hypothesis k_{hyp}. In correspondence with the key hypothesis that maximizes the correlation coefficient it is found the secret key k_r^*;

4. A single instance of CPA applied on a leakage model derived from the $S\text{-}Box(\cdot)$ allows to extract a sub-key k_r^* (a share from the actual secret key k^*). By repeating

a CPA attack from point 3. by cycling through all the indexes r of all S-Boxes, it is possible to recollect all the sub-keys k_r^* and combine them into k^*.

A key hypothesis k_{hyp} is wrong if ρ approximates 0 for each sample point l. While, a key hypothesis k_{hyp} is correct if there exists at least a point in ρ that tends to ± 1.

When considering a CPA attack, it must also be made some assumptions on the behaviour of the noise. Noise is considered random, independent from the intermediate state, and Gaussian. The smaller is the variance of the noise σ^2, the closer the Pearson's correlation coefficient will be to ± 1 under the assumption of correct key hypothesis.

When the reference state R is not known, it could still be possible to extract the correct key by using the partial correlation coefficient $\rho_{part} = \rho\sqrt{\frac{l}{m}}$, where l is the number of bits of the reference R that are known, or that could be found by exhaustive search; m is the total number of bits of the reference state R. Then, the partial correlation coefficient would allow to observe the value of the correlation coefficient ρ attenuated by the factor $\sqrt{\frac{l}{m}}$.

An optimization that could be performed for the computation of $\rho_{k_{hyp},l}$ involves storing explicitly sums of variables. This allows to perform live updates of the correlation factor ρ by updating such variables with fresh measurements:

$$\rho_{k_{hyp},l} = \frac{N\sum_{n=1}^{N} h_{n,k_{hyp}} t_{n,l} - \sum_{n=1}^{N} h_{n,k_{hyp}} \sum_{n=1}^{N} t_{n,l}}{\sqrt{\left((\sum_{n=1}^{N} h_{n,k_{hyp}})^2 - N\sum_{n=1}^{N} h_{n,k_{hyp}}^2\right)\left((\sum_{n=1}^{N} t_{n,l})^2 - N\sum_{n=1}^{N} t_{n,l}^2\right)}}$$

This algorithm is also known as *Incremental* Pearson coefficient[3], and it requires to store five variables defined as:

$$s_1 = \sum_{n=1}^{N} h_{n,k_{hyp}} \qquad s_2 = \sum_{n=1}^{N} h_{n,k_{hyp}}^2$$
$$s_3 = \sum_{n=1}^{N} t_{n,l} \qquad s_4 = \sum_{n=1}^{N} t_{n,l}^2$$
$$s_5 = \sum_{n=1}^{N} h_{n,k_{hyp}} t_{n,l}$$

That can be easily updated by simply adding new data. The correlation coefficient may be re-written as:

$$\rho = \frac{Ns_5 - s_1 s_3}{\sqrt{((s_1)^2 - Ns_2)((s_3)^2 - Ns_4)}}$$

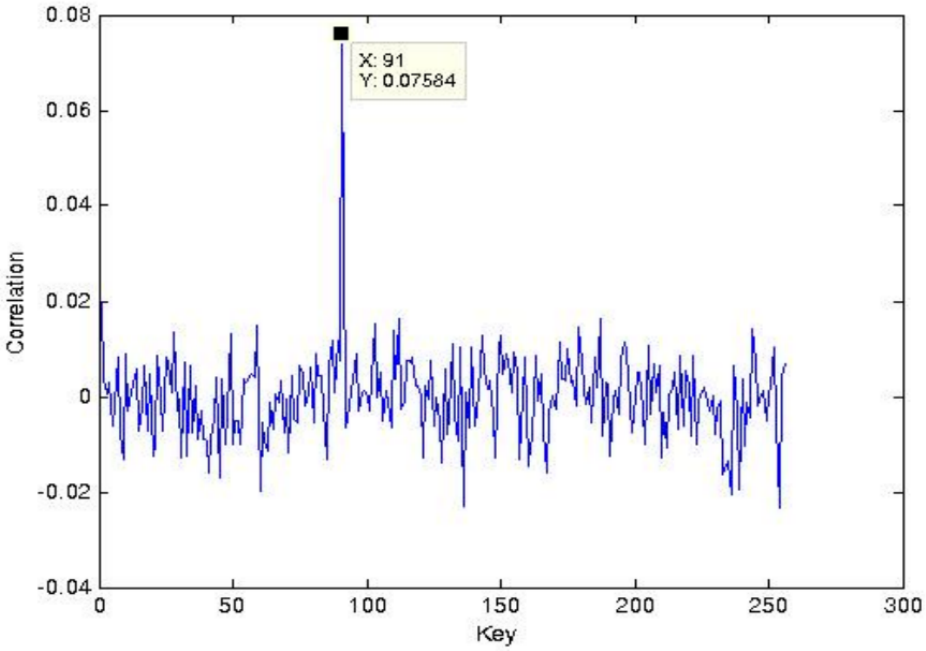


Figure 2.6: Correlation values for CPA

In figure 2.6 it can be clearly observed the correlation value for a correct sub-key guess with respect to wrong key hypothesis. The attacked algorithms is AES, and the leakage model is defined after a Substitution Box: thus in a single CPA attack it is tested 256 different sub-key values (since S-Boxes are implemented as 8-to-8 bits functions).

For what concerns countermeasures, masking is one of the most effective countermeasure that can be implemented on a device to protect it against (simple) CPA: such countermeasure is able to defeat power analysis since it masks any intermediate value, and it hinders the ability of an attacker to predict such bit values[4, 14]. As a result, the attacker must gather way more measurements from target device, and sometimes such big number of measurements may be unfeasible to be gathered. However, as it happens for high order DPAs, they cannot prevent high-order CPA attacks[13, 20, 32]. As a consequence, masking alone is not enough; masking must be combined with heterogeneous countermeasure to raise the level of security.[30]

2.4. Mutual Information Analysis

Developed in 2008 by Batina *et al.*, *Mutual Information Analysis* is the first generic differential side-channel attack of its genre[10]. Its core idea is to identify if there is any dependence between two random variables by measuring their *mutual information*. In fact, mutual information is able to measure any statistical dependence between two

random variables X and Y. Which means it is able to quantify (in terms of bits) the amount of information it is obtained by one random variable after observing the other one. Mutual information is defined as:

$$\mathbf{I}(\mathbf{X};\mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}) \tag{2.7}$$

Where $H(\mathbf{X})$ is the *Shennon entropy* of $\mathbf{X}$. The Shennon entropy is defined as the measure of the uncertainty of a random variable $\mathbf{X}$ on a discrete space $\mathcal{X}$ during an experiment:

$$H(\mathbf{X}) = -\sum_{x\in\mathbf{X}} \mathbb{P}_{\mathbf{X}}[\mathbf{X}=x]\, log_2\, \mathbb{P}_{\mathbf{X}}[\mathbf{X}=x], \tag{2.8}$$

$$\mathbb{P}_{\mathbf{X}}[\mathbf{X}=x] = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{x_i=x} \tag{2.9}$$

Where $H(\mathbf{X})$ expresses the uncertainty in bits (due to the base 2 of the logarithm); $\mathbb{P}_{\mathbf{X}}(\cdot)$ is computed by counting all the instances of possible values $x \in \mathcal{X}$. The function $\mathbb{1}_A$ is the indicator function of A: it is equal to 1 when A is true, 0 if false.

The joint entropy of two random variables $(\mathbf{X},\mathbf{Y})$ expresses the uncertainty about both random variables and it is defined as:

$$H(\mathbf{X},\mathbf{Y}) = -\sum_{x\in\mathbf{X},y\in\mathbf{Y}} \mathbb{P}_{\mathbf{X},\mathbf{Y}}[\mathbf{X}=x,\mathbf{Y}=y]\, log_2\, \mathbb{P}_{\mathbf{X},\mathbf{Y}}[\mathbf{X}=x,\mathbf{Y}=y], \tag{2.10}$$

$$\mathbb{P}_{\mathbf{X},\mathbf{Y}}[\mathbf{X}=x,\mathbf{Y}=y] = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{x_i=x,y_i=y} \tag{2.11}$$

The joint entropy is the largest when the two random variables are independent, and it decreases by the quantity $I(\mathbf{X},\mathbf{Y})$ as both random variables influence one another. The conditional entropy $H(\mathbf{X}|\mathbf{Y})$ is the measure of uncertainty of a random variable $\mathbf{X}$ on a discrete space $\mathcal{X}$ as a measure of its uncertainty during an experiment given that the random variable $\mathbf{Y}$ is known[24]. It is defined as:

$$H(\mathbf{X}|\mathbf{Y}) = -\sum_{x\in\mathbf{X},y\in\mathbf{Y}} \mathbb{P}_{\mathbf{X},\mathbf{Y}}[\mathbf{X}=x,\mathbf{Y}=y]\, log_2\, \mathbb{P}_{\mathbf{X},\mathbf{Y}}[\mathbf{X}=x|\mathbf{Y}=y], \tag{2.12}$$

$$\mathbb{P}_{\mathbf{X},\mathbf{Y}}[\mathbf{X}=x|\mathbf{Y}=y] = \frac{\sum_{i=1}^{N} \mathbb{1}_{x_i=x,y_i=y}}{\sum_{i=1}^{N} \mathbb{1}_{y_i=y}} = \frac{\mathbb{P}_{\mathbf{X},\mathbf{Y}}[\mathbf{X}=x,\mathbf{Y}=y]}{\mathbb{P}_{\mathbf{Y}}[\mathbf{Y}=x]} \tag{2.13}$$

Then, the mutual information is used as a distinguisher between the leakage model and side-channel measurements. Due to the generic nature of the attack, the leakage model is not needed to be as detailed as possible. This means that it is no more necessary to model the exact power consumption of the targeted device: knowing the cryptographic algorithm is already enough for the definition of the leakage model. In fact, the generic property of the attack comes from the distinguisher itself. Mutual information is able to capture linear, non linear, univariate, and multivariate relationships between random variables[24] (in this case hypothetical leakage model and observed leakages). But this huge advantage over the choice of the leakage model comes with a trade off in terms of traces needed for a successful attack. The attack goes as follows:

1. The attacker gathers N side-channel measurements from the target device along with the relative plaintext/ciphertext values in vectors $\mathbf{T}^n[0..L]$ and $\mathbf{M}^n$ respectively. The traces come from several execution of a known cryptographic algorithm under the same and unknown secret key k^*.

$$\mathbf{T}^n[0..L] = \begin{bmatrix} \mathbf{t}^1 \\ \mathbf{t}^2 \\ \vdots \\ \mathbf{t}^N \end{bmatrix} = \begin{bmatrix} t_0^1 & t_1^1 & \cdots & t_L^1 \\ t_0^2 & t_1^2 & \cdots & t_L^2 \\ \vdots & \vdots & \ddots & \vdots \\ t_0^N & t_1^N & \cdots & t_L^N \end{bmatrix}$$

,

$$\mathbf{M}^n = \begin{bmatrix} m^1 \\ m^2 \\ \vdots \\ m^N \end{bmatrix}, \quad 0 \leq m \leq M$$

In general, measurements are saved into a matrix $\mathbf{pmf}[T][M]$ as:

$$\mathbf{pmf}[t][m] = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,M} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,M} \\ \vdots & \vdots & \ddots & \vdots \\ p_{T,0} & p_{T,1} & \cdots & p_{T,M} \end{bmatrix}, \ 0 \leq t \leq T, \ 0 \leq m \leq M$$

Where M is the total number of possible plaintexts/ciphertexts that can be fed/ gathered from the cryptographic algorithm, and T is the set of distinct discrete values that could be observed from side-channel measurements. If the values of T are not discrete, a preprocessing step must be performed on these value T. Then, for each occurrence of a value t, it is incremented the count by 1 of the variable $p_{t,m}$, a variable indexed by the value t itself and the relative plaintext/ciphertext that is being handled. In block ciphers, plaintext/ciphertexts are divided in R blocks of equal size. Thus, it can be built R matrixes (for each plaintext/ciphertext block) $\mathbf{pmf}_r[T][M_r]$ as:

$$\mathbf{pmf}_r[t][m] = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,M_r} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,M_r} \\ \vdots & \vdots & \ddots & \vdots \\ p_{T,0} & p_{T,1} & \cdots & p_{T,M_r} \end{bmatrix}, \ 0 \leq t \leq T, \ 0 \leq m \leq M_r,$$

$$m^n = \{m_1^n, ..., m_r^n, ..., m_R^n\}, \quad 0 \leq m_r \leq M_r, \quad M_r << M$$

2. Then it is defined the hypothetical leakage model of the target device. The model is defined after an intermediate value computed from the cryptographic algorithm. In literature, it is chosen as target a S-Box computation, as it involves the application of both secret key and known data. A hypothetical leakage model is defined as $\hat{L}_{k_{hyp}} := f(m, k_{hyp}) = S\text{-}Box(m, k_{hyp})$, where the îdentifies the hypothetical nature of the model that differs from the real and unknown leakage model $L_{k_{hyp}}$. Since the real leakage model cannot be known, but it can be found a model that is close to the real one, it will be used interchangeably "hypothetical leakage model" and "leakage model" unless it is stated differently.

   If it is targeted a block cipher, it means that secret keys will be divided as well into

R blocks (as many as plaintext/ciphertext blocks). Thus, it will be implemented R (possibly different) S-Boxes: a model $\hat{L}$ defined after such *S-Box* will then allow to retrieve a single sub-key k_r^*;

3. It is first fixed the index r of the attacked *S-Box*$(\cdot)$ and the key hypothesis k_{hyp}. Being $\mathbf{T}$ a random variable representing all the possible (discrete) values that could be observed from side-channel measurements, it is estimated both $\mathbb{P}_{\mathbf{T}}$ and $\mathbb{P}_{\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}}$. $\mathbb{P}_{\mathbf{T}}$ is the distribution of observed measurements; $\mathbb{P}_{\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}}$ is the distribution of measurements given the random variable of the hypothetical leakage model $\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}$ under key hypothesis k_{hyp}. These values are used for the computation of the mutual information through the formula:

$$I(\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}; \mathbf{T}) = H(\mathbf{T}) - H(\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}) \tag{2.14}$$

Where $I(\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}; \mathbf{T})$ is the mutual information between the two random variables; $H(\mathbf{T})$ is the entropy of random variable $\mathbf{T}$; $H(\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}})$ is the conditional entropy of random variable $\mathbf{T}$ given $\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}$.

The way those two distributions are built is with the histograms technique. The distribution $\mathbb{P}_{\mathbf{T}}$ is obtained from the matrix $\mathbf{pmf}_r[t][m]$ by reducing the matrix to a column vector, obtained by gathering all the rows values into a unique variable $p_t = \sum_{i=1}^{M_r} p_{t,i}$:

$$\mathbb{P}_{\mathbf{T}} \approx \mathbf{distr}_{samples}[t] = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_T \end{bmatrix}, \ 0 \leq t \leq T \tag{2.15}$$

While the conditional distribution $\mathbb{P}_{\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}}$ is built from the matrix $\mathbf{pmf}_r[t][m]$ by first computing the joint distribution
$\mathbf{distr}_{joint}[t][y]$. The variable y and distributions are defined as:

$$y = \hat{L}_{k_{hyp}}(m) = \textit{S-Box}(k_{hyp} \oplus m) \tag{2.16}$$

$$\mathbb{P}_{\mathbf{T},\mathbf{Y}} \approx \mathbf{distr}_{joint}[t][y] = \mathbf{pmf}_r[t][y] \tag{2.17}$$

$$\mathbb{P}_{\mathbf{Y}} \approx \mathbf{distr}_{hyp}[y] = \sum_{i=1}^{T} \mathbf{distr}_{joint}[i][y] \tag{2.18}$$

$$\mathbb{P}_{\mathbf{T}|\mathbf{Y}} \approx \mathbf{distr}_{cond}[t][y] = \frac{\mathbf{distr}_{joint}[t][y]}{\mathbf{distr}_y[y]} \tag{2.19}$$

And the mutual information is computed with (2.14), by first plugging (2.15) into

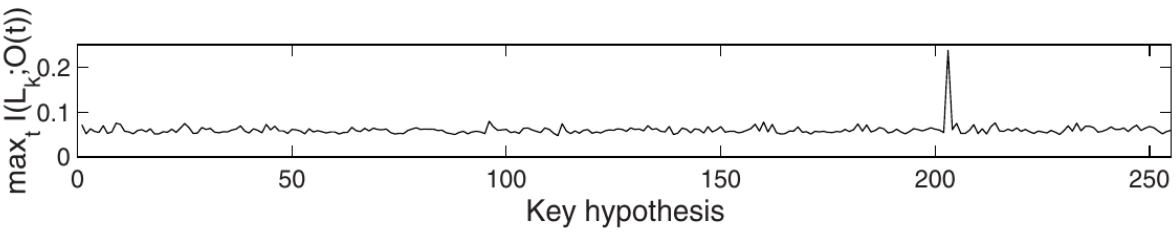


Figure 2.7: Mutual information values of MIA

(2.8), and then by plugging (2.17) and (2.19) into (2.12) respectively;

4. It is repeated the third step for each key hypothesis k_{hyp}; in correspondence with the maximum value of the mutual information it is identified the secret key k^*:

$$I(\hat{\mathbf{L}}_{\mathbf{k}^*}; \mathbf{T}) = arg\, max_{k_{hyp}} I(\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}; \mathbf{T}) \tag{2.20}$$

5. If the leakage model is defined after a S-Box computation, then a single instance of MIA will extract one single sub-key of the whole secret key k^*. Then, the attack must be repeated for each $S\text{-}Box(\cdot)$ (fed with different key material) by repeating the attack from step 3. with different values for r;

6. Then it is repeated MIA for each sample point l. At each sample point l, the mutual information will be approximate the value '0' if there is no relation between secret keys tested with the hypothetical leakage model and side-channel measurements. Then, it will be observed significant values only in those sample points of side-channel measurements where there is actual information about the secret sub-key.

In figure 2.7 it is shown the values of mutual information for each key hypothesis. The attacked algorithm is an AES implementation, and as leakage model it is chosen the 3MSB of a $S\text{-}Box(\cdot)$ output, that is $\hat{L}_{k_{hyp}} = S\text{-}Box(m \oplus k_{hyp})[5:7]$.

Due to the generality of mutual information, hiding countermeasures alone are not enough for data protection, since gathering and averaging a lot of measurements is enough to pursuit a successful attack.

In addition to it, as stated before, MIA is able to extract univariate, multivariate, linear, and non-linear dependencies between observed measurements and models. Thus, by adjusting the definition of the entropy to consider multiple points in side-channel measurements, it could be enhanced the attack to high-order mutual information analysis. This latter attack, is capable to extract a secret key even if protected with masking[2].

Thus, masking alone is not enough, and additional countermeasures must be implemented for data protection.

2.4.1. Properties of the Leakage Model

The choice of the leakage model $\hat{L}$ must meet some important requirements[10]. Let L be a leakage model: it can be defined the set $\{L_0, L_1, ..., L_n\}$ as subset of the space $\mathcal{L}$, the space of all possible leakage values, where each element L_i is called atom. E.g. if the leakage model is the Hamming Weight of a S-Box output, then $\mathcal{L} = \{0, 1, 2, ..., f\}$, where f is the number of bit used to represent the result of a S-Box output. Otherwise, if as leakage model it is chosen the cryptographic function computing the S-Box output, then it would be $\mathcal{L} = \{L_0, L_1, ..., L_{2^f}\}$. By defining a partition of $\mathcal{L}$ as the set of elements $\{L_0^{k_{hyp}}, ..., L_{2^f}^{k_{hyp}}\}$ under key hypothesis k_{hyp}, each element is defined as:

$$L_i^{k_{hyp}} = \{m \in \{0,1\}^b \mid \hat{L}(\mathbf{W}_{k_{hyp}}) = i \ \wedge \ \mathbf{W}_{k_{hyp}} = (v_1, f_{k_{hyp}}(m))\}, \qquad (2.21)$$
$$0 \leq i \leq 2^f, \ 0 \leq k_{hyp} \leq K, \ 0 \leq b \leq B$$

Where $m \in \mathbf{M}$ are all the possible values for the plaintext/ciphertext message; b is the number of bits used to represent the message; $\hat{L}$ is the user defined leakage model; the random variable $\mathbf{W}_{k_{hyp}}$ represents the occurrence of two words within the execution of the cryptographic algorithm, namely a constant (possibly unknown) reference state v_1, and the attacked cryptographic function $f_{k_{hyp}}(m)$. The latter one will depend on both secret key hypothesis and plaintext/ciphertext value.

The above formula 2.21 associates all the input values $\mathbf{M} = m$ that leak $\hat{\mathbf{L}}_{k_{hyp}} = i$ under key hypothesis k_{hyp} to the atom $L_i^{k_{hyp}}$. That is, it partitions all the set of plaintexts/ciphertexts values with respect to the outcome of the leakage model fed with a particular key hypothesis k_{hyp} and message m. This classification of plaintext/ciphertext values induces a classification of side-channel measurements, since each side-channel measurement is associated with one value of plaintext/ciphertext.

If the cryptographic function used for the model (in this case the S-Box) is a bijective function, (that is, there is a one-to-one correspondence between inputs and outputs), then mutual information will not be able to distinguish the secret sub-key. This problem is caused by the injective property of the function, since for each input there exists only one output. If a leakage model is defined after an f-to-f bits S-Box, then there will be 2^f inputs that will be associated with one of the 2^f outputs.

By choosing as leakage model the *Identity* $Id(\cdot)$ of the substitution box output $L_{k_{hyp}} :=$

$Id(\text{S-Box}(k_{hyp}, m)) = \text{S-Box}(k_{hyp}, m)$, and having fixed a key hypothesis k_{hyp_1}, such leakage model will define a partition of plaintext/ciphertext values to atoms $L_i^{k_{hyp_1}}$. By changing key hypothesis k_{hyp_2}, the exact same plaintext/ciphertext values that were previously associated with $L_i^{k_{hyp_1}}$, will be now associated with another atom $L_i^{k_{hyp_2}}$. What actually happens is that for each key hypothesis the same batch of plaintext/ciphertext values will be associated to atoms $L_i^{k_{hyp}}$.

Then, by changing values of key hypothesis, what is actually changed is the "label" used to represent that particular group of plaintext/ciphertext values. Since what changes is only the label, changing key hypothesis will not affect the computation of mutual information, that will be equal for each hypothesis. Thus it will not allow to discern the true key from the wrong ones.

As workaround it is defined as leakage model as a subset of the output of a cryptographic function $\hat{L}_{k_{hyp}} := \text{Subspace}(\text{S-Box}(k_{hyp}, m)) = \text{S-Box}(k_{hyp}, m)[c : v]$, where $[c : v]$ defines a bit range. This workaround, also known as *bit dropping* strategy, does allow to relax the injective property of the target function. By relaxing the injectivity, each output will be associated with more than one inputs: for each key hypothesis, each output it will be then associated with different inputs.

This would mean that it is possible to associate to each atom $L_i^{k_{hyp}}$ different groups of plaintexts/ciphertexts with respect to key hypothesis k_{hyp}. In turn, the computation of the mutual information will be different for each key hypothesis, as it is considered a distinct groups of plaintexts/ciphertext (and thus, distinct groups of side-channel measurements).

In 2014, Gierlichs et al.[25] wrote a paper about some properties of the leakage model for generic side-channel attacks (like MIA). It was noticed that leakage models defined by means of the bit dropping strategy applied to bijective target functions were good characterization of a target device (as they would lead to successful SCAs).

However, for what concerns MIA, side-channel attacks that were carried out with leakage models defined as $\hat{L}_{k_{hyp}} := \text{drop}(\text{S-Box}(k_{hyp}, m))_b$ for bits $b \in \{2, 3, 4, 5, 6, 7\}$ were successful. On the other hand, when the leakage model was defined as $\hat{L}_{k_{hyp}} := \text{drop}(\text{S-Box}(k_{hyp}, m))_1$, it failed unexpectedly.

This phenomenon was noticed when MIA was tested with ideal measurements, in absence of noise ($\sigma_n = 0$). In fact it was observed that if the standard deviation σ_n of the noise would reasonably increase (above a certain threshold), MIA would eventually succeed. Thus, only in high-SNR scenario, the bit drop would not work.

Moreover, it was observed that for MIA, a leakage model defined by dropping 5 bits

$\hat{L}_{k_{hyp}} := drop(S\text{-}Box(k_{hyp}, m))_5$, would lead to a higher success rate with respect to other bit dropping combinations with the same number of measurements. The reason why is thought to be a superposition of two effects[25]:

- As the number of dropped bits increases, the target becomes more "non-injective", and thus the attack would work better;
- From a number of dropped bits from 5 to 7, the algorithmic noise of the dropped bits start to become predominant, worsening the performance of the attack.

In the same paper, it was also observed that it could be defined a leakage model $\hat{L}_{k_{hyp}}$ without the need of dropping bits. Instead of attacking directly a $S\text{-}Box(\cdot)$, it could be also targeted deeper operations in a cryptographic round, and it could be possibly identify a non-injective target.

Such operations would ideally perform a compression of secret key and known data. In the paper, it was picked out a *MixColumns* operation of an AES encryption. The first byte computed by a *MixColumn* is computed by mixing four input bytes (u, v, w, x) as $z = 2u \oplus 3v \oplus w \oplus x$.

As leakage model it could be chosen to target a partial intermediate value $z = 2u \oplus 3v = 2S\text{-}Box(k_1 \oplus m_1) \oplus 3S\text{-}Box(k_6 \oplus m_6)$, of which would be considered the entire output.

It was shown that a MIA attack performed with model z, would lead to a successful attack by testing 2^{16} different values for the couple of keys (k_1, k_6).

2.4.2. Probability Distribution Estimation

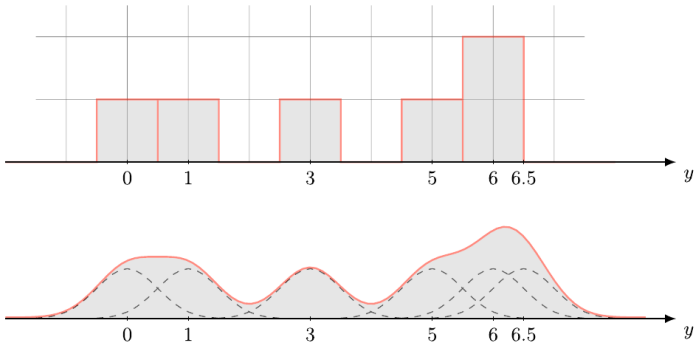


Figure 2.8: Histogram and Kernel method respectively

The estimation of probability distributions $\mathbb{P}_{\mathbf{T}}$ and $\mathbb{P}_{\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}}$ is one important topic since all the computations depend on them.

In the paper where MIA was introduced[10], it was chosen to use the *Histogram Method*: this method makes use of several bins, and estimates the probability distribution of data in a sample set by counting how many samples fall into a certain bin.

However there is no optimal strategy that can offer the best estimation. In [10] it was observed that by using as many bins as the total number of distinct side-channel values that can be observed, it is possible to exploit as much as possible the information present in measurements. On the other hand, this approach would require way more measurements so as to fill all the bins.

Less bins will generally imply less information. However, if the measurements are very noisy, it was observed in [10] that less bins may have the effect of noise reduction. This would mean that by increasing the bins size (and reducing the amount of bins), it would be possible to classify correctly measurements that may come from the same datum.

In 2018 it was presented a paper by Chérisey et al.[7] were it was observed that the best binning size strongly depends on the standard deviation of the noise. The smaller the standard deviation of the noise, the more bins should be used to extract as much information as possible. The higher the standard deviation, the smaller must be the number of the bins.

Another estimating tool that can be used in alternative to histograms is the *Kernel* method[2]. A kernel is a function characterized by a shape and a bandwidth. For each sample, instead of classifying it in a bin of a histogram, it is added to the estimated distribution a small kernel centered on the value of the computed leakage[2], as it is shown in figure 2.8. The sum of all these small functions results to be equal to the estimation of the target distribution. It was also observed that kernels converge faster than histograms towards the actual true estimation, and are more suitable when estimating continuous distributions.

2.4.3. Optimizations

Algorithm 3: Fast computation algorithm of MIA

Data: $\mathbf{T}^n$ N measurements that take discrete values, $t^n \in \mathcal{T}$, $\mathbf{M}^n$ N plaintext/ciphertexts, $m \in \mathcal{M}$

Result: $I(\mathbf{t}^n, \mathbf{y}(k))_{k \in \mathcal{K}}$

for $i \in \{1, ..., N\}$ **do**
  $\mathbf{pmf}[t^i][m^i] \leftarrow \mathbf{pmf}[t^i][m^i] + 1$;
end
for $t \in \mathcal{T}$ **do**
  $\mathbf{distr}_{samples}[t] \leftarrow 0$;
  for $m \in \mathcal{M}$ **do**
    $\mathbf{distr}_{samples}[t] \leftarrow \mathbf{distr}_{samples}[t] + \mathbf{pmf}[t][m]$;
  end
end
for $k \in \mathcal{K}$ **do**
  $\forall\ t \in \mathcal{T},\ y \in \mathcal{Y},\ \mathbf{distr}_{joint}[t][y] \leftarrow 0$;
  for $m \in \mathcal{M}$ **do**
    for $t \in \mathcal{T}$ **do**
      $\mathbf{distr}_{joint}[t][\phi(f(k,m))] \leftarrow \mathbf{distr}_{joint}[t][\phi(f(k,m))] + \mathbf{pmf}[t][m]$;
    end
  end
  $I(\mathbf{T}^n, \mathbf{y}(k)) \leftarrow 0$;
  for $y \in \mathcal{Y}$ **do**
    $\mathbf{distr}_{hyp}[y] \leftarrow 0$;
    for $t \in \mathcal{T}$ **do**
      $\mathbf{distr}_{hyp}[y] \leftarrow \mathbf{distr}_{hyp}[y] + \mathbf{distr}_{joint}[t][y]$;
    end
    for $t \in \mathcal{T}$ **do**
      if $\boldsymbol{distr}_{samples}[t] \neq 0$ ***and*** $\boldsymbol{distr}_{hyp}[y] \neq 0$ **then**
        $I(\mathbf{T}^n, \mathbf{y}(k)) \leftarrow$
        $I(\mathbf{T}^n, \mathbf{y}(k)) + \frac{\mathbf{distr}_{joint}[t][y]}{N} log_2\Big(\frac{N\mathbf{distr}_{joint}[t][y]}{\mathbf{distr}_{samples}[t]\mathbf{distr}_{hyp}[y]}\Big)$
      end
    end
  end
end
return $\mathcal{I}(\mathbf{T}^n, \mathbf{y}(k))_{k \in \mathcal{K}}$

In the paper of Chérisey et al. [7] it was also proposed a series of optimizations that could be performed on MIA for a faster computation of the overall attack.

The leakage model can be defined as $\hat{L}_{k_{hyp}} := \phi(f(m, k_{hyp})) = \mathbf{y}(k_{hyp})$. $f(\cdot)$ is an intermediate value of the cryptographic implementation; $\phi(\cdot)$ is a function chosen with some (possibly partial) knowledge of leakage model; both $f(\cdot)$ and $\phi(\cdot)$ are chosen such that they fulfill the *Markov condition*[7]. The Markov condition states that the leakage $\mathbf{t}$ depends on the secret key k^* only through the computed model $y = \phi(f(k^*, m))$.

The computation of the probability mass function $pmf[T][M]$ allows to classify side-channel measurements with respect to the plaintext messages associated to them. Measurements $\mathbf{T}^n$ do already take discrete values, thus they can be classified into the matrix $pmf[T][M]$ by checking in which bin samples fall in.

Then it is computed the distribution of measurements $\mathbf{distr}_{samples}[T]$ that will be used for the computation of the mutual information $I(\cdot)$.

For each key hypothesis it must be computed the joint distribution matrix $\mathbf{distr}_{joint}[T][Y]$, indexed by means of discrete measurements values and hypothetical leakage model estimations under key hypothesis k_{hyp}.

Then it is optimized the computation of the mutual information analysis: for each index t, it first computed anew the distribution of values induced by the leakage model, and then it is updated the mutual information with new material.

The computation of the mutual information of this algorithm allows to perform $|\mathcal{T}| * |\mathcal{Y}|$ $log_2(\cdot)$ computations (that are the most expensive operations that could be performed from a computational point of view). With respect on how the space $|\mathcal{T}|$ is defined by means of the histogram method, and on how the leakage model space $|\mathcal{Y}|$ is defined by the attacker, it can be performed a relatively fast computation of MIA.

2.5. Comparison between CPA and MIA

A year after the introduction of MIA, it was presented a paper that evaluated the performance of MIA and CPA under Gaussian assumption [21]. It was shown that both CPA and MIA have similar performance for the same number of side-channel measurements when it used a power model that approximates the real leakage function of the target device, e.g., Hamming Weight (or Hamming Distance). However, when it is used an intermediate value of a cryptographic algorithm as leakage function, it was noticed that CPA achieved better results than MIA at discerning the secret key (for the same number

of plaintexts).

In fact, in 2018 Heuser et al. [12] have shown that CPA attack is the optimal attack that can be performed when the model of a target device is known and when the noise is Gaussian, that is when $N \sim \mathcal{N}(0, \sigma^2)$.

However, in 2018 it was presented by Chérisei et al. a paper that compared MIA with CPA [7] in two distinct scenarios that were particularly challenging for a CPA attack. From the study[7] it was shown that:

- MIA outperforms CPA when the leakage model of the target device is known, while the noise does not follow the Gaussian assumption;
- MIA outperforms CPA when the leakage model is unknown (or partially known), with Gaussian noise.

In the first case, the reason why MIA turned out to be better than CPA was due to the fact that CPA needed a noise that followed a Gaussian assumption for a successful attack. While, it was observed that MIA performance were not affected by the type of noise, as the variance of the noise did not influence the mutual information estimation.

In the second case, it was observed that MIA was capable to outperform CPA when the leakage model was a non-linear function of the Hamming Weight, even if the noise was Gaussian. In fact, CPA resulted to be optimal when the leakage model was linear; on the other hand, mutual information was capable to extract any relation between a (suitable) general leakage model and side-channel measurements.

The test was performed by testing the same models (assuming the device leaked information as the hamming weight model), and tested the same amount of side-channel measurements. As figure of merit used to compare CPA and MIA was the *Success Rate*. The Success Rate (or SR), is defined as the total number of successful attacks, carried out for each cryptographic encryption computed with a distinct secret key value, averaged by the total number of secret key values[7]:

$$SR = \frac{1}{2^n} \sum_{k=0}^{2^n-1} \mathbb{P}_k(k^* = k) \tag{2.22}$$

This kind of success rate is also known as *first-order* success rate, as it is considered the first key that is pointed out form the distinguisher as the most probable for the evaluation of the efficiency of an attack. However, it can be also considered success rates of higher order, e.g. second-order success rate, where it is considered also if an attack is successful

with the second most probable key (in addition to the first one). This metric is useful when there are two keys that are estimated with the same score from a distinguishing tool; if one of these two keys is the secret key, there will be probability $P \sim \frac{1}{2}$ of choosing the right key for the attack.

Another metric that can be used to describe the effectiveness of an attack is the *Guessing Entropy*[19]. The guessing entropy (GE) can be defined as the average position of the correct key in a vector of all possible keys (ranked from the most probable to the least probable). Then, the higher the position of the correct key in the vector, the better are the performances of an attack.

It was also shown that MIA tends asymptotically (as the number of measurement increase) to the *Maximum Likelihood* expression in those two distinct case of study. In fact, the Maximum Likelihood distinguisher is the best distinguisher that can be implemented capable to maximizes the success rate of a side-channel attack. Since there is no profiling phase, the leakage probabilities are replaced with online estimated probabilities[7] computed at the moment, right after the sampling phase.

The Maximum Likelihood key estimation expression is defined as:

$$k^* = arg\ max_{k_{hyp}} \tilde{\mathbb{P}}(\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}) = arg\ max_{k_{hyp}} I(\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}), \tag{2.23}$$

$$\tilde{\mathbb{P}}(\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}}) = \prod_{i=1}^{N} \tilde{\mathbb{P}}(\mathbf{t}^i|y_i) = \prod_{i=1}^{N} \tilde{\mathbb{P}}(\mathbf{t}^i|y(k, m^i)) = \prod_{i=1}^{N} \tilde{\mathbb{P}}\big(\mathbf{t}^i|\phi(f(k_{hyp}, m^i))\big) \tag{2.24}$$

Where $I(\cdot)$ is the mutual information value, and the distribution $\tilde{\mathbb{P}}(\cdot)$ is computed online. Thus the maximum likelihood searches for that key hypothesis that allows to maximize the conditional distribution. maximizing the distribution $\tilde{\mathbb{P}}(\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}})$ is equal to minimize the conditional entropy $H(\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}})$; in turn it will maximize the mutual information expression (2.7), because $H(\mathbf{T})$) is constant, and the maximum of mutual information is achieved when $H(\mathbf{T}|\hat{\mathbf{L}}_{\mathbf{k_{hyp}}})$ is the smallest possible.

A recent research trend in countermeasures against SCAs did delve into the development of alternative ways to increase the resiliency of a cryptographic cipher to side-channel attacks. Since the main target of SCAs are Substitution Boxes outputs, researchers started to study and develop S-Boxes resilient to side-channel attacks.

In 2017, Lerman et al.[16] developed several S-Boxes resilient to CPAs and TAs (*Template Attacks*). Such S-Boxes would require an attacker to gather an increasing number of

measurements in order to pursuit a successful side-channel attack.

The experiment was carried out by searching for small sized S-Boxes (5x5 bits S-Boxes and 4x4 bits S-Boxes) with genetic algorithms. It was generated and attacked a pool of S-Boxes, which were then filtered by means of a fitness function that promoted those S-Boxes that minimised the overall success rate. Then, the study compared the performance between S-Boxes generated with genetic algorithms and S-Boxes actually implemented in real applications, and noticed that genetic algorithms were capable to derive S-Boxes that were more resilient to SCAs than the implemented ones.

The improved resiliency required the attacker to gather more side-channel measurements from a target device. The way it was observed that generated substitution boxes were better was to perform several attacks with always an increasing number of measurements, and save the relative success rate. Then, it was found generated substitution boxes had overall a lower success rate when comparing attacks with small traces.

As these tests were performed with a SNR higher than 1 ($\sim 2.x$), it was observed that in a real scenario, a small reduction of the success rate on a substitution box would require more measurements in order to perform a successful attack with an acceptable success rate for that specific substitution box.

From this point on, researchers studied several substitution boxes, looking for non linear functions with particular characteristics, like resiliency to high-order attacks[15].

3 | Leakage Model Exploration

The objective of this work is to explore leakage models built over several distinct intermediate functions, and identify those functions resilient to MIA when it is used as a distinguisher the maximum of mutual information. In this process, it is investigated the success rtae of MIA attacks on leakage models that are the closest possible to the actual leaking behaviour of the target device.

In addition to it, if it can be identified such functions (and leakage models as a consequence), it is then investigated if it could be also found in a systematic way leakage models that are the furthest possible to the leaking behaviour of the target device. If this investigation is successful, then those leakage models would point out the correct secret key when it is used the minimum of mutual information as key distinguisher.

In this chapter it will be explained how the experiments were set up and how the whole investigation was carried out.

3.1. Leakage Model Definition

The investigation of the best leakage models $\hat{L}_{k_{hyp}}$ with respect to a suitable cryptographic function $f(\cdot)$ was performed with the application of two non fixed masks $mask_{in}$ and $mask_{out}$ to arguments in input of $f(\cdot)$ and at the output of $f(\cdot)$ respectively. Thus, for a suitable choice for $f(\cdot)$, the leakage model would be defined as:

$$\hat{L}_{k_{hyp}} := mask_{out} \,\&\, f(m \,\&\, mask_{in}, k_{hyp}) \tag{3.1}$$

The $mask_{in}$ variable allows to explore combinations of inputs to the function $f(\cdot)$; in turn, $mask_{in}$ allows to explore combinations of output values computed by the target function itself. If the function $f(\cdot)$ is a non-linear function, then $mask_{in}$ will be able to control combinations of non-linear outputs. If the function is bijective, varying the $mask_{in}$ variable would not affect the bijective property, since it could be seen as a way to limit the amount of inputs fed to $f(\cdot)$.

On the other hand, $mask_{out}$ variable allows to explore bits combinations of the outputs of $f(\cdot)$. If $f(\cdot)$ is a bijective function, then varying the output mask will break the bijective property (by breaking the injective property) by compressing inputs values into output values. In turn it would allow to compute a correct estimation of the mutual information. The function $f(\cdot)$ would still be surjective since each output value computed form function $f(\cdot)$ will be associated to at least one input value.

Being $f(\cdot)$ a d-to-g bits function, the number of distinct values taken from the masks will be $0 \leq mask_{in} \leq 2^d$ and $0 \leq mask_{out} \leq 2^g$. If $d \neq g$, then the function $f(\cdot)$ would be a non-injective, surjective (non bijective) function. Leakage models built on such functions would lead to successful attacks with MIA if the variations of $mask_{in}$ and $mask_{out}$ would retain the non-injective property of the function.

If $d = g$, then function $f(\cdot)$ would be an injective, surjective, and thus bijective function. By varying $mask_{in}$, the function would still be bijective; on the other hand, varying $mask_{out}$ would allow to obtain a non-injective function (and thus, non-bijective), and it lead to a correct estimation of the mutual information.

In general, a leakage model would lead to a successful attack if it is built from a non-injective d-to-g bits function, with $d > g$.

As intermediate functions used for the construction of leakage models, it is chosen the substitution boxes of distinct block ciphers. Since block ciphers encrypt messages by computing several rounds of cryptographic operations, it is possible to dissect the interested cryptographic operation out of the entire computation.

This setup would allow to test in a fast and efficient way only substitution boxes computation without computing additional overhead. In the studied block ciphers, substitution boxes perform a non-linear substitution of the value computed as the logical XOR operation between a block of plaintext messages and a block of secret key of the same length.

Then, the leakage model for a MIA attack is defined as:

$$\hat{L}_{k_{hyp}} := mask_{out} \ \& \ \textit{S-Box}((m \ \& \ mask_{in}) \oplus k_{hyp}) \tag{3.2}$$

While, the leakage model for a CPA attack is defined as:

$$\hat{L}_{k_{hyp}} := HW\big(mask_{out} \ \& \ \textit{S-Box}((m \ \& \ mask_{in}) \oplus k_{hyp})\big) \tag{3.3}$$

For the experiment, it was first chosen to analyze the substitution box of AES block

cipher, as it is a 8-to-8 bits substitution box.

It is also analyzed a substitution box that it is non-injective by nature, as the first substitution box of DES cipher. In fact it is a 6-to-4 bits substitution box, which is clearly asymmetric. Its analysis allows to confirm that by using the identity model as leakage model, it does lead to a successful attack when attacked with MIA.

Then, in line with Lerman et al.[16], it is tested leakage models built from the 4x4-bits substitution boxes of known ciphers, as PRESENT, PRINCE and Klein, and leakage models built from substitution boxes generated with genetic algorithm identified in [16]. The same analysis is performed with 5x5 bits S-Boxes of renowned ciphers, as ASCON and PRIMATE, and S-Boxes found in [16].

3.2. Side-Channel Attacks Implementations

For this project, it was implemented via software both CPA and MIA side-channel attacks. It was chosen to encode both attacks with C programming language. It was chosen the C programming language simply due to performance reasons. While, the analysis of results was performed with Python because it offers an environment rich of fast and performant analysis tools.

The metric used to test both attacks is the success rate. For each leakage model driven by the combination of variables $mask_{in}$ and $mask_{out}$, it is performed an encryption for each secret key, and each encryption is attacked with both CPA and MIA. Then, each successful attack will be recorded for the computation of the success rate for that particular leakage model.

Each side-channel attacks will be performed with the same number of side-channel measurements (an thus, plaintext values). CPA is known to need less measurements for a successful attack[22], but it is not the objective of this work to find the minimum amount of traces from which a side-channel attack does fail or it does succeed for a particular leakage model. Instead, it is chosen an amount of plaintexts values that are enough for both side-channel attacks to be successful.

For each distinct parameter d of d-to-g bits substitution boxes, it is generated a set of plaintext values made of 10000 records. This set of plaintexts is generated by drawing an equal number of distinct values for plaintexts in the range $[0, 2^d]$. Then, for each distinct plaintext, there will be around $\textbf{integer}(\frac{10000}{2^d}) + p$ representatives. The function $\textbf{integer}(\cdot)$ takes only the integer part of the result of the computation of the division operation. The value p will be equal to 1 if a plaintext value $ptxt$ is smaller than $10000 -$

$\textbf{integer}(\frac{10000}{2^d}) * 2^d$; otherwise p will be equal to 0.

3.2.1. Mutual Information Analysis Implementation

It was chosen to implement the Mutual Information Analysis attack by encoding the pseudo-code **Algorithm 3** presented in section 2.4.

However, the algorithm must be adapted to the new definition of the leakage model and to the computation of synthetic measurements as in **Algorithm 4** and **Algorithm 5**.

For each value of the variable $mask_{in}$, it is computed a vector $m[N]$ of N plaintext values multiplied with the logical & operator with $mask_{in}$.

The functions $\textbf{max}(\cdot)$ and $\textbf{min}(\cdot)$ return the maximum and minimum values of a set of side-channel measurements, respectively. Once it is identified the entire range of measurements, it is then divided into ranges of equal size, where each range is associated with a distinct bin.

For each $mask_{out}$, and each profile of SNR, it is computed via the histogram method the probability mass function matrix $pmf[BIN][M]$.

Then, each side-channel measurement $t[N]$ is classified into bins with respect to the range it falls in.

As each side-channel measurement gets classified into a particular bin, it is updated the counter of the matrix $pmf[BIN][M]$ in correspondence with the bin the sample got classified into, and the relative plaintext $m[N]$ associated to such sample.

Before the test of key hypothesis k_{hyp}, it is first computed the distribution of output measurements $\textbf{distr}_{samples}[BIN]$ (as the sample distribution is kept equal for each key hypothesis test).

Finally, for each key hypothesis k_{hyp}, and for each possible value of messages m and (binned) measurements t, it is computed the joint distribution $\textbf{distr}_{joint}[BIN][Y]$. The space $\mathcal{Y}$ includes all the possible values computed by the hypothetical leakage model.

From the joint distribution matrix $\textbf{distr}_{joint}[BIN][Y]$ it is possible to compute the distribution of values induced by the leakage model $\textbf{distr}_{hyp}[Y]$.

For the current project it has been investigated a particular bin size strategy. This strategy relies on choosing a number of bins (with equal width), which is equal to the number of discernible values that can be obtained from the leakage model. If a leakage model is built from a d-to-g function, it will be linked to the parameter g as the following definition:

Algorithm 4: Estimation of Distribution with Histograms

Data: $\mathbf{T}^n$ N discrete measurements, $t^n \in \mathcal{T}$, $\mathbf{M}^n$ N plaintext / ciphertexts, $m \in \mathcal{M}$, $BINS$ number of bins

Result: Estimation of distribution probabilities

for $mask_{in} \in \{1, .., 2^g\}$ **do**
  $m[1, .., N] \leftarrow \{0\}$;
  for $i \in \{1, .., N\}$ **do**
    $m[i] \leftarrow m^i$ & $mask_{in}$
  end
  for $mask_{out} \in \{1, .., 2^d\}$ **do**
    $pmf[BIN][M] \leftarrow \{0\}$;
    $sample_{max}, sample_{min} \leftarrow \mathbf{max}(t[1, .., N]), \mathbf{min}(t[1, .., N])$;
    $range \leftarrow (sample_{max} - sample_{min})/BINS$;
    for $j \in \{1, .., N\}$ **do**
      for $i \in \{1, .., BIN\}$ **do**
        if
        $(t[j] >= sample_{min} + (i-1) * range)$ ***and*** $(t[j] < sample_{min} + i * range)$
        then
          $pmf[i][m[j]] \leftarrow pmf[i][m[j]] + 1$;
        end
      end
    end
    $\mathbf{distr}_{samples}[T] \leftarrow \{0\}$;
    for $m \in \mathcal{M}$ **do**
      $\mathbf{distr}_{samples}[t] \leftarrow \mathbf{distr}_{sample}[t] + pmf[t][m]$;
    end
    for $k_{hyp} \in \mathcal{K}$ **do**
      $\mathbf{distr}_{joint}[BIN][M] \leftarrow \{0\}$;
      for $y \in \mathcal{Y}$ **do**
        for $t \in \{1, .., BIN\}$ **do**
          $\mathbf{distr}_{joint}[t][y] \leftarrow \mathbf{distr}_{joint}[t][y] + pmf[t][y]$;
        end
      end
      $\mathbf{distr}_{hyp}[Y] \leftarrow \{0\}$;
      for $t \in \{1, .., BIN\}$ **do**
        $\mathbf{distr}_{hyp}[y] \leftarrow \mathbf{distr}_{hyp}[y] + \mathbf{distr}_{joint}[t][y]$
      end
    end
  end
end

$$BINS := 2^{HW(g\ \&mask_{out})} \tag{3.4}$$

This strategy was actually used in the first paper of MIA[10], and it will be referred in the work as *batina_ bins*. This kind of binning method would only makes use of information about how many distinct values can be captured by the leakage model.

Another strategy that is used in research fixes the number of bins to 256, as it simulates a real sampling process that quantizes side-channel measurements. Usually, oscilloscope used for the sampling process have eight bits of resolution. Thus, once it is fixed a range referred to the order of the target measurements, it can be represented at most 256 distinct values (as with 8 bits there will be at most $2^8 = 256$ values).

Since there are 256 distinct values that could be observed, 256 bins would allow to classify each side-channel measurement into a specific bin that is used only for that side-channel value. This kind of choice does not imply any assumption on the target device, and aims at exploiting as much as possible the information carried out from side-channel measurements.

However, it can also be used more expensive instruments with higher resolution, as 12-bits or 16-bits resolutions, at the cost of an increasing effort from a computational point of view, and it does require an increasing number of measurements.

Another strategy that is also used in research relies on dividing the side-channel measurements range into a small set of bins of equal size[22, 7]. As cited already in section 2.4, this choice might actually help in the estimation of distributions with noisy measurements. This strategy does also boost the overall computation of the mutual information.

A final strategy used to choose a suitable number of bins for the estimation of probability distribution relies on *Scott's Rule*[27] and Freedman-Diaconis rule[9]:

$$BINS := 3.49 * \sigma(T) * N^{\frac{-1}{3}} \tag{3.5}$$

$$BINS := 2 * IRQ(T) * N^{\frac{-1}{3}} \tag{3.6}$$

Where $\sigma(T)$ is the standard deviation of observations, N is the total amount of measurements, IRQ is the interquartile range.

The choice of using *batina_ bins* does allow to implement a hybrid strategy that computes the mutual information faster with small values for the variable $mask_{out}$, and then perform

more fine-grained computations for bigger values of $mask_{out}$.

The computation of mutual information is computed in the same fashion of algorithm (3):

Algorithm 5: Mutual Information Estimation

Data: $\mathbf{T}^n$ N discrete measurements, $t^n \in \mathcal{T}$, $\mathbf{M}^n$ N plaintext / ciphertexts, $m \in \mathcal{M}$, $BINS$ number of bins
Result: $I(\mathbf{T}^n, \mathbf{y}(k))_{k \in \mathcal{K}}$
for $mask_{in} \in$ **do**
  for $mask_{out} \in$ **do**
    for $k^* \in \mathcal{K}$ **do**
      for $k \in \mathcal{K}$ **do**
        $\vdots$
        $I(\mathbf{T}^n, \mathbf{y}(k)) \leftarrow 0$;
        for $i \in \{1, .., BIN\}$ **do**
          for $y \in \mathcal{Y}$ **do**
            if $\boldsymbol{distr}_{samp}[i] \neq 0$ ***and*** $\boldsymbol{distr}_{hyp}[y] \neq 0$ **then**

$$I(\mathbf{T}^n, \mathbf{y}(k)) \leftarrow I(\mathbf{T}^n, \mathbf{y}(k)) + \frac{\mathbf{distr}_{joint}[i][y]}{N} log_2\Big(\frac{N\mathbf{distr}_{joint}[i][y]}{\mathbf{distr}_{samp}[i]\mathbf{distr}_{hyp}[y]}\Big)$$

            end
          end
        end
      end
    end
  end
end
return $I(\mathbf{T}^n, \mathbf{y}(k))$

3.2.2. Correlation Power Analysis Implementation

Algorithm 6: CPA implementation

Data: $\mathbf{T}^n$ N measurements that take discrete values, $t^n \in \mathcal{T}$, $\mathbf{M}^n$ N plaintext/ciphertexts, $m \in \mathcal{M}$

Result: $\rho(\mathbf{T}^n, k)_{k \in \mathcal{K}}$

//If *S-Box*$(\cdot)$ is a d-to-g bitbits function **for** $mask_{in} \in \{1, ..., 2^d\}$ **do**
  for $mask_{out} \in \{1, ..., 2^g\}$ **do**
    for $k^* \in \mathcal{K}$ **do**
      //It is fixed a secret key
      for $k_{hyp} \in \mathcal{K}$ **do**
        //Key guesses
        $\mu_t \leftarrow 0$;
        for $i \in \{1, .., N\}$ **do**
          $\mu_t \leftarrow \mu_t + t^i$;
        end
        $\mu_t \leftarrow \frac{\mu_t}{N}$;
        $\mu_{h_{hyp}} \leftarrow 0$;
        $h_{hyp}[1, .., N] \leftarrow (0, .., 0)$;
        for $i \in \{1, .., N\}$ **do**
          $h_{hyp}[i] \leftarrow HW(mask_{out}$ & *S-Box*$(k_{hyp} \oplus (m$ & $mask_{in})))$;
          $h_{hyp} \leftarrow h_{hyp} + h_{hyp}[i]$;
        end
        $h_{hyp} \leftarrow \frac{h_{hyp}}{N}$;
        $sum_{num}, sum_{den1}, sum_{den2} \leftarrow 0, 0, 0$;
        for $i \in \{1, .., N\}$ **do**
          $t^i \leftarrow t^i - \mu_t$;
          $h_{hyp}[i] \leftarrow h_{hyp}[i] - \mu_{h_{hyp}}$;
          $sum_{num} \leftarrow sum_{num} + t^i h_{hyp}[i]$;
          $sum_{den1} \leftarrow sum_{den1} + t^i t^i$;
          $sum_{den2} \leftarrow sum_{den2} + h_{hyp}[i] h_{hyp}[i]$;
        end
        $\rho_{k,\mathbf{T}} = \frac{sum_{num}}{\sqrt{sum_{den1} sum_{den2}}}$;
      end
    end
  end
end

For what concerns CPA, it was chosen to encode the unoptimized CPA attack presented in section 2.3.

The only difference with the optimized attack is in how fast the coefficient ρ is computed. Since the pool of measurements chosen for the experiments is relatively small, the overall computations will not be greatly affected by this choice.

The leakage models used for CPA attacks were defined by assuming that side-channel measurements leaked information by following the Hamming Weight model (or Hamming Distance model with null reference $R = 0$).

The computation of ρ is not affected in the least whether the function $f(\cdot)$ is bijective or not. Nevertheless, it is chosen to explore leakage models in the same way it is explored for MIA simply for symmetric reasons.

3.3. Side-Channel Measurements Simulation

Side-channel measurements used for the experiment were built synthetically within the code. For each side-channel attack (both CPA and MIA), it was chosen to simulate a device that leaked information in its power consumption under the Hamming Weight model. For the experiment, it is chosen to store one single value of power consumption in correspondence of the computation of the substitution boxes.

CPA and MIA are then used to attack leakage models built from Substitution Boxes $S\text{-}Box(\cdot)$ by using these synthetic measurements. Thus, side-channel measurements are simply made of one sample, and they can be defined as a one dimensional vector:

$$\mathbf{T}^n = \begin{bmatrix} \mathbf{t}^1 \\ \mathbf{t}^2 \\ \vdots \\ \mathbf{t}^N \end{bmatrix} = \begin{bmatrix} t^1 \\ t^2 \\ \vdots \\ t^N \end{bmatrix} \tag{3.7}$$

Side-channel measurements are computed anew for each side-channel attack on a cryptographic (sub-)key.

Algorithm 7: Computation of ideal side-channel measurements

Data: k^* secret key, $\mathbf{M}^n$ N plaintexts (possibly with a mask applied on)
Result: artificial set of N measurements **T**
$T[1,..,N] \leftarrow (0,..,0)$;
for $i \in \{1,..,N\}$ **do**
    $T[i] \leftarrow HW(S\text{-}Box(m^i \oplus k^*))$;
end
return T

It is first fixed the values of the masks $mask_{in}$ and $mask_{out}$ of the leakage model. Then, it is fixed the value of the secret key k^*, and it is computed the $S\text{-}Box(\cdot)$ function for each plaintext value m^n. For each S-Box computation, it is stored the corresponding Hamming Weight values into a vector indexed by the values of plaintexts m^n as shown in procedure 7.

For the first experiment it was chosen to test both side-channel attacks to ideal synthetic measurements, thus with zero noise.

Then it was chosen to test both side-channel attacks in different settings of *Signal-to-Noise Ratio*. The SNR is defined as the ratio between the variance of a signal and the variance of the noise:

$$SNR := 10 * log_{10}\Big(\frac{Var(signal)}{Var(noise)}\Big) = 10 * log_{10}\left(\frac{\Big(\sum_{i=1}^{N} \frac{t^i - \mu_t}{N}\Big)^2}{(\sigma_{noise})^2}\right) dB \tag{3.8}$$

Where μ_t is the average value between all the synthetic measurements, and $(\sigma_{nosie})^2$ is the variance of the noise. The signal, in this case, it is represented by the power consumption measurements in correspondence with the computation of the substitution boxes. Then SNR is used to measure the quality of a signal with respect to the noise. The higher is the ratio, the better the quality of the signal. The smaller is the ratio, the worse is the quality of the signal with respect to the noise.

It was chosen to test different SNR profiles as they would allow to study which is the best leakage model that should be employed for a SCA with respect to that particular SNR profile.

The simulated noise was produced by a method described by Abramowitz and Stegun[1, 29]:

Algorithm 8: noise_generator$(\cdot)$ procedure: draws random samples from a Gaussian distribution with 0 mean and standard deviation equal to 1

Data: *null*
Result: *Output*
static U, V;
static $P \leftarrow 0$;
$Output \leftarrow 0$;
if P ***is equal to*** 0 **then**
    $U = \frac{(rand() + 1.0)}{RAND\_MAX + 2.0}$;
    $V = \frac{rand()}{RAND\_MAX + 1.0}$;
    $Output = \sqrt{-2log(U)}sin(2\pi V)$;
else
    $Output = \sqrt{-2log(U)}cos(2\pi V)$;
end
$P \leftarrow 1 - P$;
return *Output*

Where the static variables value P, U, V are preserved at each call of the random Gaussian number generator. The values taken by U fall within the range $]0, +1[$, excluding both 0 and +1. While, the values of V fall within the range $[0, +1[$, 0 included and +1 excluded. The domain of $sin(\cdot)$ and $cos(\cdot)$ are in the range $[-1, +1]$; the values of $log(U)$, with respect to how it is defined $RAND\_MAX$, is within range $[log(1/(RAND\_MAX + 2)), log((RAND\_MAX + 1)/(RAND\_MAX + 2))] \approx [-9.331929866, 0)$ (if $RAND\_MAX = 2147483647$ as it was in the C implementation of the project).

At the first call, it is executed the statements in the body of **if**. Thus, it is computed anew U, V, and $Output$. At the second run, it is computed the statement in the body of **else**. Then, the following calls of the function alternate the execution of statements between the bodies of **if** and **else**.

This function is used to produce Gaussian noise with zero mean, and standard deviation equal to 1, which means samples are drawn from distribution $N \sim \mathcal{N}(0, \sigma_{noise}) \sim \mathcal{N}(0, 1)$. To simulate other profiles of standard deviation of the noise, it could be performed by simply multiplying the standard deviation σ_{noise} with the target standard deviation $\sigma_{target} = \sigma_{noise}\sigma_{target}$.

Since it was wanted to generate the same noise succession values for each independent execution, it is initialized the random function generator with the same seed. This would

allow to compare different side-channel attacks on the same synthetic measurements.

By choosing a suitable value for the SNR, it is computed the standard deviation of the noise as:

$$\frac{SNR}{10} = log_{10}\Big(\frac{Var(signal)}{Var(noise_{target})}\Big) \equiv 10^{\frac{SNR}{10}} = \Big(\frac{Var(signal)}{Var(noise_{target})}\Big) \Rightarrow \tag{3.9}$$

$$Var(noise_{target}) = \frac{Var(signal)}{10^{\frac{SNR}{10}}} \Rightarrow \sigma_{target} = \sqrt{\frac{Var(signal)}{10^{\frac{SNR}{10}}}} \tag{3.10}$$

While the variance of the signal $Var(signal)$ is computed anew for each combination of variables $mask_{in}, mask_{out}$, and k^* from the noise-free measurements samples.

By plugging a suitable for the SNR in 3.10, it is then computed the value of the standard deviation σ_{target} that can be multiplied to noise samples so as to simulate noisy measurements. Such measurements would still be with zero mean, but they would have updated their standard deviation $N \sim \mathcal{N}(0, \sigma_{target})$.

It follows the SNR profiles chosen for the study:

- SNR ratios of 10dB hypothesize an attacker that possess expensive and very accurate tools for the acquisition of side-channel measurements. In fact, the attacker would be capable to obtain high quality measurements;
- SNR ratio of 1dB hypothesize an attacker that owns decent tools for side-channel measurements. He would be capable of retrieving side-channel measurements with a ratio with respect to noise bigger than 1;
- SNR ratio of -10dB hypothesize an attacker with cheap tools for the acquisition of measurements. He would sample bad quality measurements where the standard deviation of the noise is predominant with respect to the standard deviation of the signal. Thus, the measurements is predominantly made of noise.

Then the side-channel measurements computation procedure is updated with the addition of Gaussian random noise:

Algorithm 9: compute_measurements$(\cdot)$ procedure: computes side-channel measurements with Gaussian noise

Data: k^* secret key, $\mathbf{M}^n$ N plaintexts (possibly with a mask applied on), a value for SNR $\in \{-10, 1, 10\}dB$

Result: artificial set of N measurements **T**

$T[1, .., N] \leftarrow (0, .., 0)$;

for $i \in \{1, .., N\}$ **do**

  $T[i] \leftarrow HW(S\text{-}Box(m^i \oplus k^*))$;

end

$\mu_t \leftarrow 0$;

for $i \in \{1, .., N\}$ **do**

  $\mu_t \leftarrow \mu_t + T[i]$;

end

$\mu_t \leftarrow \frac{\mu_t}{N}$;

$var_t \leftarrow 0$;

for $i \in \{1, .., N\}$ **do**

  $var_t \leftarrow var_t + (T[i] - \mu_t)^2$;

end

$var_t \leftarrow \frac{var_t}{N}$;

$\sigma_{target} \leftarrow \sqrt{\frac{var_t}{10^{\frac{SNR}{10}}}}$;

for $i \in \{1, .., N\}$ **do**

  $T[i] \leftarrow T[i] + \textbf{filter}(\textbf{noise\_generator}())\sigma_{target}$;

end

return T

Where the **filter**$(\cdot)$ function caps the values generated by the noise generator to values within the range $[-3, +3]$. Since the Gaussian function generates values as a normal distribution with 0 mean and 1 standard deviation, calls to the function **noise_generator**$(\cdot)$ would to draw values within the range $[-1, +1]$ 68.27% of the time.

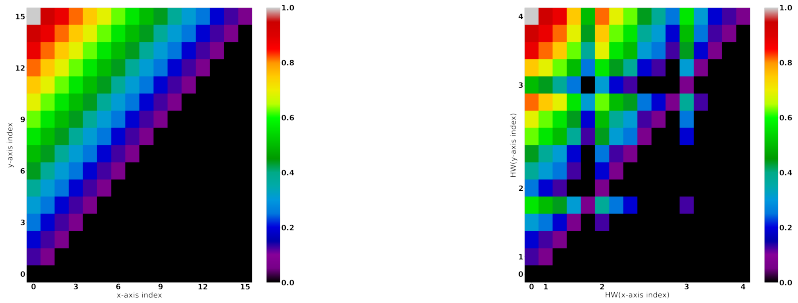
Values between the range $[-2, +2]$ are drawn 95.45% of the time, and values within the range $[-3, +3]$ are drawn 99.73% of the time.

Capping the values exceeding the range $[-3, +3]$ would only discard the 0.27% of the information of some outlier values that would not be seen with real measurement instruments. In fact, real side-channel measurements (related to an encryption with fixed values for m and k^*), get averaged together several times. If any of those outlier values would

appear, their effect would get canceled through the averaging process. In addition to it, such outlier values would impair the estimation of distributions as they would wrongly identify the highest and/or the smallest values of side-channel measurements.

3.4. Data Visualization

For each attacked substitution box it is built a heat-map of success rates, where on the horizontal axis it is reported the hamming weight of the variable $mask_{out}$, while on the vertical axis it is reported the hamming weight of the variable $mask_{in}$. Since for each distinct hamming weight value there will possibly be multiple permutations of bits within a binary string $mask$ that lead to same hamming weight value, it is chosen to order the various model that are equal under the same hamming weight perspective in increasing order.



(a) A heat-map indexed with ordinal values

(b) A heat-map indexed with the hamming weight of ordinal values

For e.g., in the heat-map of figure (3.1b), it is shown a 16x16 cells cluster heat map, where both x-axis and y-axis are ordered with respect to the hamming weight of the indexes of figure (3.1a). That is, for the hamming weight equal to 1, it will be aligned in increasing order the y-axis values along the indexes $\{1, 2, 4, 8\} = \{< 0001 >_2, < 0010 >_2, < 0100 >_2, < 1000 >_2\}$. The vertical bar conveys the success rate value, associating to each color shade a success rate value. It was chosen this way for data representation as it was observed that success rate shew interesting patterns with respect to hamming weights of variables $mask_{in}$ and $mask_{out}$.

4 | Experimental Evaluation

In this chapter it will be observed the main results of the experiments, and it will be given a comment about them.

It will first be observed the behaviour of leakage models built from the substitution box of AES cipher, which characteristics were already observed in literature[31, 33], and it were given some probabilistic interpretation on some leakage models [33].

Then it will be verified if the identity model does lead to a successful attack with MIA when it is used a substitution box that it is non-injective by nature, such as the first S-Box of DES cipher.

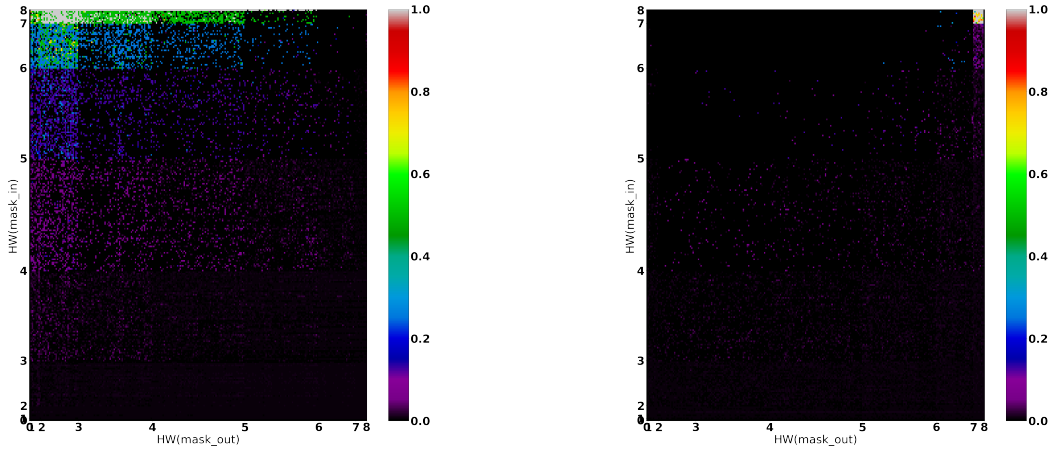
Finally, it will be observed the behaviour of leakage models built over 4x4 bits and 5x5 bits substitution boxes.

4.1. AES Substitution Box

In the paper presented at *CRYPTO 2011* by Standaert et al.[31] it was shown that the estimations of MIA attacks of leakage models $\hat{L}_{k_{hyp}} := \textit{S-Box}(m \oplus k_{hyp})_{1:n}$ built from the substitution box of AES cipher, with $1 \leq n \leq 7$, revealed that the mutual information estimation was the minimum in correspondence to the secret key when 7-bits models and noiseless measurements were used.

Then it was first chosen to analyze all the leakage models built from AES' S-Box, and observe if it could be found some patterns.

It follows the heatmap of success rates of MIA attacks on leakage models built from AES' S-Box (by tweaking $mask_{in}$ and $mask_{out}$ variables):



(a) SR on AES's S-Box with max MI

(b) SR on AES's S-Box with min MI

Figure 4.1: SR on AES's S-Box with noiseless measurements

It can be observed that the highest success rates are observed when leakage models are built from hamming weight values of variable $mask_{in}$ are equal to 8, (that is $HW(mask_{in})$ = 8), and for several hamming weight values of $mask_{out}$ (that is, when $HW(mask_{out}) = \{1, 2, 3, 4, 5\}$). For some leakage models where $HW(mask_{out}) = 5$ there are some cases where the attack fails. From $HW(mask_{out}) = \{6, 7, 8\}$ there is no leakage model that does lead to a successful attack.

It can also be noticed that leakage model build with combinations of $mask_{out}$ and $mask_{in}$ values where $HW(mask_{out}) = 2$ and $HW(mask_{in}) = \{7, 8\}$, it is observed a higher success rate range with respect to other models. It can be noticed that MIA fails in correspondence of any value of $mask_{in}$ and for $HW(mask_{out}) = 8$, which in turns reflects what it was expected in theory.

On the other hand, when it is used as a distinguisher the minimum of mutual information, it could be found that hypothetical leakage models $\hat{L}_{k_{hyp}}$ made with binary combinations of variable $mask_{out}$ such that $HW(mask_{out}) = 7$ and $HW(mask_{in}) = \{7, 8\}$, it leads to side-channel attacks with high success rate. As always, in correspondence of $HW(mask_{out}) = 8$ the attack fails, as it is expected in theory.

However, as soon as the side-channel measurements get spoiled with noise, even if the SNR of samples approximates to 10dB, it could not be possible to identify any leakage model exploitable with the minimum of mutual information:

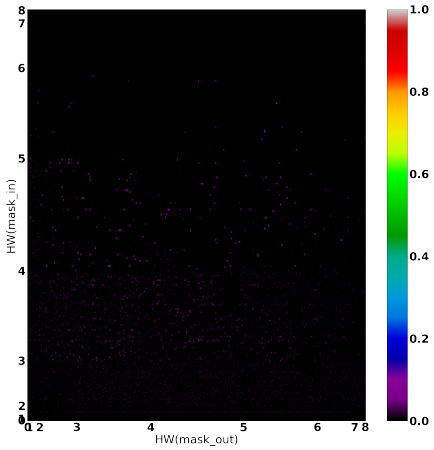
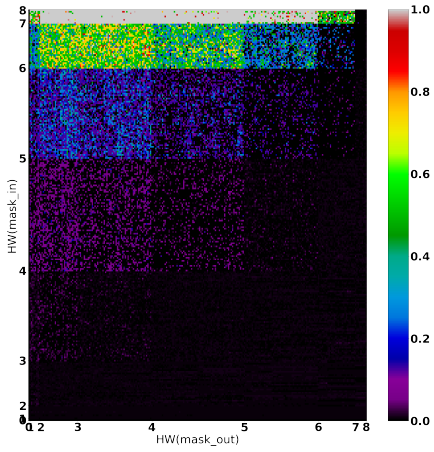
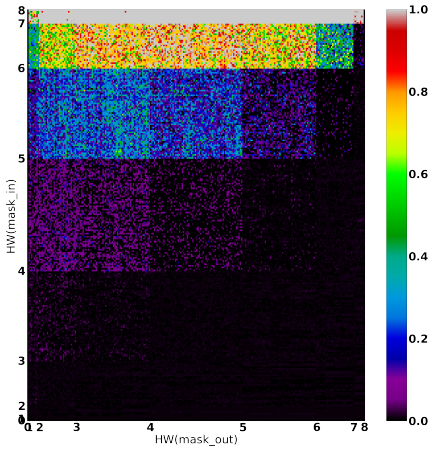


Figure 4.2: SR on AES's S-Box with min MI and 10dB SNR measurements

Once it is simulated side-channel measurements with different SNR, it was observed that only the maximum of mutual information was able to extract the secret key of a cryptographic encryption:

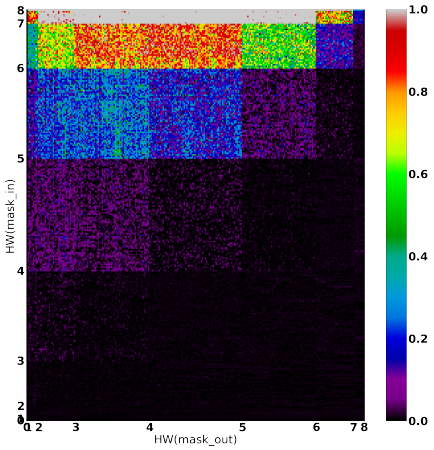


(a) SR on AES's S-Box with 10dB SNR



(b) SR on AES's S-Box with 1dB SNR

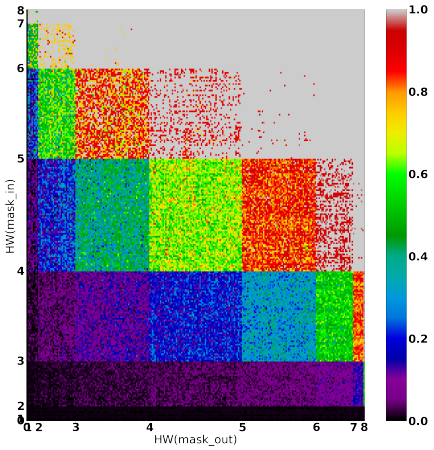
It can be noticed that when side-channel measurements have SNR equal to 1dB, it is observed the highest SR in correspondence of hypothetical leakage models with $HW(mask_{in})$ $= \{7, 8\}$ and $HW(mask_{out}) = \{1, 2, 3, 4, 5, 6, 7\}$.



(a) SR on AES's S-Box with -10dB SNR

It could be also observed MIA attacks with a SNR that approximates -10dB is more successful with respect to measurements with SNR equal to 10dB. As always, in correspondence of $HW(mask_{out}) = 8$, each MIA attack fails, as it is expected in theory.

A CPA attack on the same leakage models verifies the efficiency of CPA attacks on target devices when it is known their leaking behaviour. As it can be observed, when $HW(mask_{out}) = 8$ and $HW(mask_{in}) = 8$ CPA does lead to successful attacks since Pearson's correlation factor is not affected by the bijectivity of the S-Box:



(a) SR on AES's S-Box with CPA and noiseless measurements

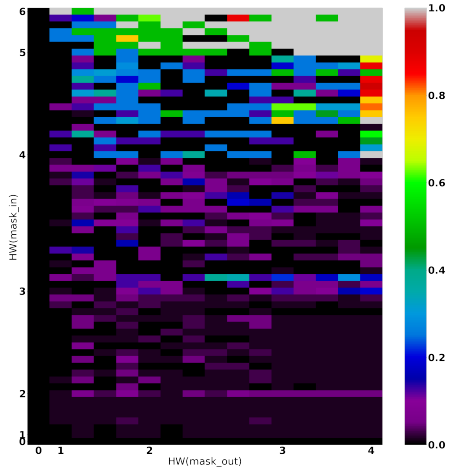
AES S-Box

Table 4.1: Success Rate with the first most probable key guess

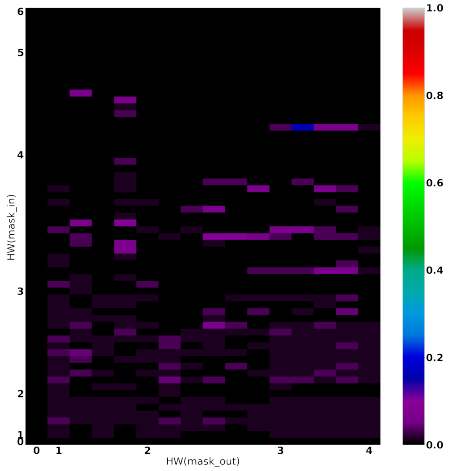
SNR	max MI	min MI
∞	✓	✓
10dB	✓	✗
1dB	✓	✗
-10dB	✓	✗

4.2. DES Substitution Box

DES cipher is made of eight distinct 6x4 substitution boxes. As all the substitution boxes are non-injective, it was chosen to analyze the first $S\text{-}Box(\cdot)$ of DES cipher. The study of this substitution box is used to verify that the identity model $\hat{L}_{k_{hyp}} := mask_{out}$ & $S\text{-}Box((m \& mask_{in}) \oplus k_{hyp}) = S\text{-}Box(m \oplus k_{hyp})$ does lead to a successful attack with MIA.



(a) SR on DES' S-Box with max MI and with noiseless measurements



(b) SR on DES' S-Box with min MI and with noiseless measurements

It can be noticed that the mutual information analysis performed by looking at the maximum of mutual information does lead to successful attacks when the leakage model is defined as the identity model ($HW(mask_{in}) = 6$ and $HW(mask_{out}) = 4$). It is also successful when $HW(mask_{in}) = \{5, 4\}$, with some degradation of performance when the hamming weight of $mask_{in}$ is equal to 4.

A high success rate is also observed when variables $mask_{in}$ have hamming weight $HW(mask_{in}) = 5$, and for some values for $mask_{out}$, where $HW(mask_{out}) = \{2, 3, 4\}$.

Then, as soon as the HW of variables $mask_{in}$ drops to 3, it will not be possible to compress different distinct inputs into one output value from the S-Box function, and thus the attack will fail. As it is seen for the AES S-Box, reducing the amounts of distinct plaintexts values (accomplished by small values for the hamming weight of $mask_{in}$ variables), worsens a lot the efficiency of the attack.

However, when it was looked for the systematically wrong leakage models, it couldn't be found a model that does lead to a successful attack with a high success rate.

As usual, CPA attack on DES' hypothetical leakage models are more effective then MIA when it is known how information is leaked from the target device:

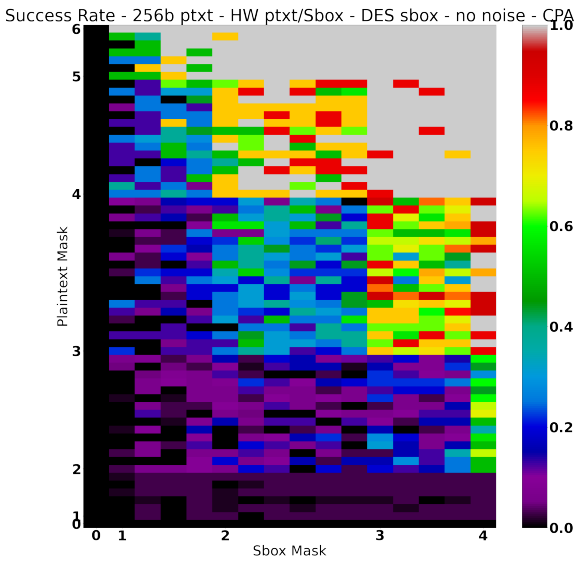


Figure 4.7: Success rate of CPA on leakage models built from DES' first S-Box

DES S-Box

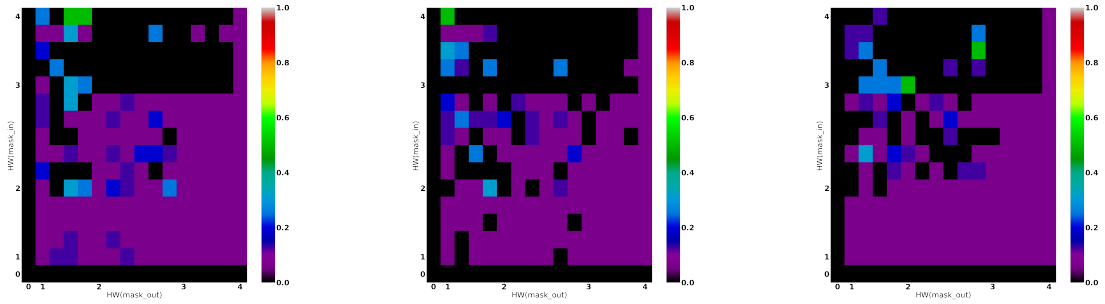
Table 4.2: Success Rate with the first most probable key guess

SNR	max MI	min MI
∞	✓	✗
10dB	✓	✗
1dB	✓	✗
-10dB	✓	✗

4.3. 4x4 Substitution Box

In this chapter it will be analyzed the results obtained when leakage models are built from 4x4-bits substitution boxes generated through genetic algorithms [16] and from 4x4-bits substitution boxes of renowned ciphers.

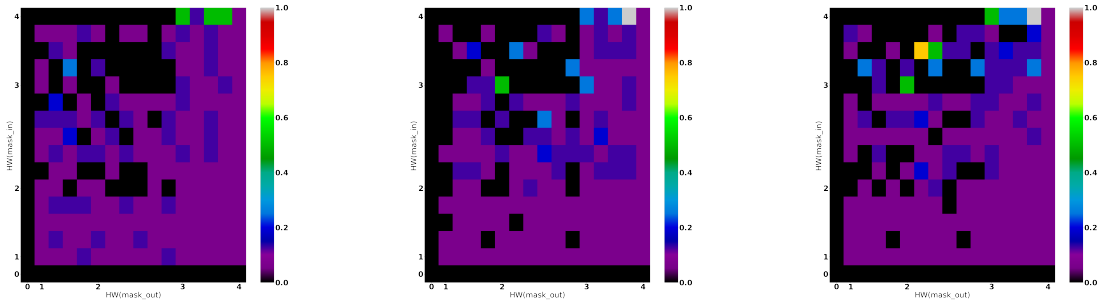
Starting from substitution boxes implemented in real world applications, it was noticed that MIA attacks performed with the maximum of mutual information as a key distinguisher would not lead to any successful attack with a satisfying success rate:



(a) SR on PRESENT's S-Box (b) SR on PRINCE's S-Box (c) SR on Klein's S-Box

Figure 4.8: SR with noiseless measurements and max MI

On the other hand, when it was used the minimum of mutual information as distinguisher, it was observed that it could be found a leakage model for both Klein and PRINCE ciphers that would lead to a successful attack with success rate equal to 1 when it was observed the minimum of mutual information.



(a) SR on PRESENT's S-Box (b) SR on PRINCE's S-Box (c) SR on Klein's S-Box

Figure 4.9: SR with noiseless measurements and min MI

It can be noticed that when it is analyzed the minimum of the mutual information estimation, only the substitution box of PRESENT cipher does not allow to find a leakage model that would retrieve a (sub-)key with success rate equal to 1 for an SNR equal to -10dB. However, when it is analyzed the second-order success rate, it can be found for PRESENT's S-Box some leakage models that are close to the leaking behaviour of the target device, also for a SNR of -10dB. Unfortunately it couldn't be found additional leakage models that were actually the furthest from he target one.

PRESENT/PRINCE/Klein S-Box

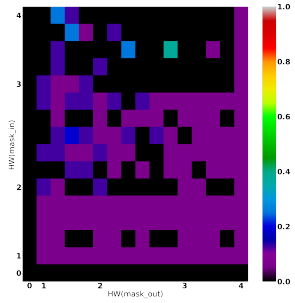
Table 4.3: Success Rate with the first most probable key guess

SNR	max MI	min MI
∞	✗/✗/✗	✗/✓/✓
10dB	✓/✗/✗	✗/✗/✓
1dB	✓/✓/✗	✗/✗/✗
-10dB	✗/✓/✓	✗/✗/✗

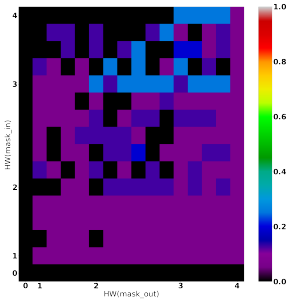
Table 4.4: Success Rate with the first two most probable key guesses

SNR	max MI	min MI
∞	✓/✓/✗	✓/✓/✓
10dB	✓/✓/✓	✗/✗/✓
1dB	✓/✓/✓	✗/✗/✗
-10dB	✓/✓/✓	✗/✗/✗

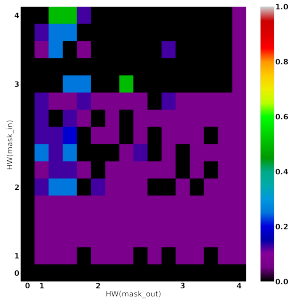
However, the two substitution boxes *evolved_to* and *evolved_cc* used as a reference functions in the paper of Lerman et al.[16] were particular resilient to MIA attacks with respect to minimum of mutual information. *Evolved_cc* was also particularly resilient to to MIA attack with respect to the maximum of mutual information as well:



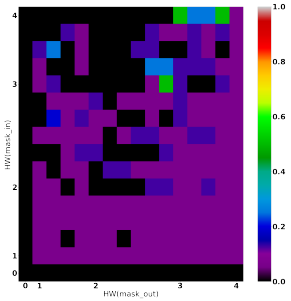
(a) SR on evolved_cc S-Box with max of MI and noiseless measurements



(b) SR on evolved_cc S-Box with min of MI and noiseless measurements



(c) SR on evolved_to S-Box with max of MI and noiseless measurements



(d) SR on evolved_to S-Box with min of MI and noiseless measurements

In particular, *evolved_cc* was the only 4x4-bits substitution box that did not offer any exploitable leakage model with the minimum of mutual information with success rate higher than ~ 0.3, and that only one leakage model was exploitable with the maximum of mutual information for a success rate of at most ~ 0.4. Also the second-order success rate wouldn't be of help on finding acceptable leakage models for *evolved_cc* substitution box.

evolved_to/evolved_cc S-Box

Table 4.5: Success Rate with the first most probable key guess

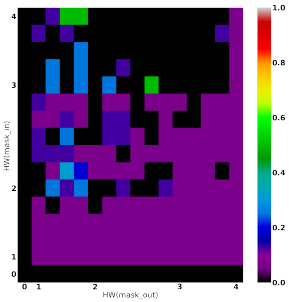
SNR	max MI	min MI
∞	✗/✗	✗/✗
10dB	✓/✗	✗/✗
1dB	✓/✗	✗/✗
-10dB	✓/✗	✗/✗

Table 4.6: Success Rate with the first two most probable key guesses

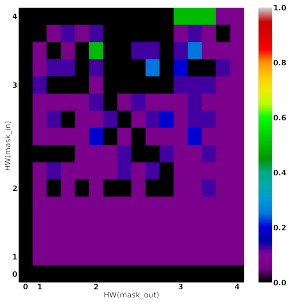
SNR	max MI	min MI
∞	✓/✗	✓/✗
10dB	✓/✗	✗/✗
1dB	✓/✗	✗/✗
-10dB	✓/✗	✗/✗

4.3.1. CPA Resilient Substitution Boxes

For what concerns the substitution boxes designed to be resilient to CPAs, it was observed that the substitution box *evolved_sr2* that was designed to be the substitution box that was the most resilient to CPAs, was also particularly resilient to MIA attacks when first-order success rate was taken into consideration:



(a) SR on evolved_sr2 S-Box with max of MI and noiseless measurements

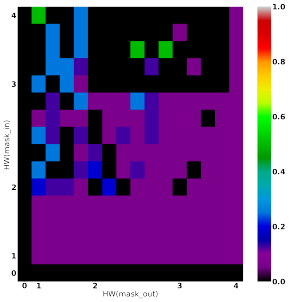


(b) SR on evolved_sr2 S-Box with min of MI and noiseless measurements

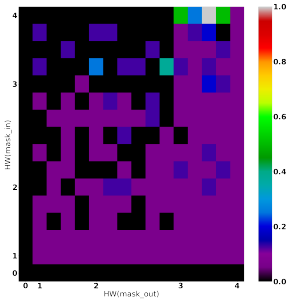
Evolved_sr1 and *evolved_k* substitution boxes, on the other hand, were both resilient to MIA attacks when it was observed the maximum of mutual information as distinguishing tool; nevertheless, it could still be found a satisfying leakage model for the minimum of mutual information.

However, second-order success rate would point out some furthest and closest leakage models for both evolved_sr1 and evolved_sr2 for 10dB SNR measurements. It was also observed that it was *evolved_k* substitution box that was in principle designed to favour

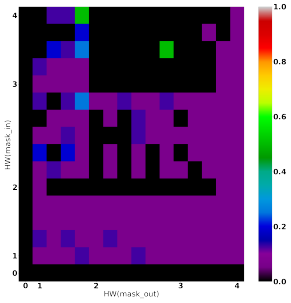
side-channel attacks that was actually the most resilient one, as it couldn't be found a furthest leakage model when measurements had 10dB of SNR.



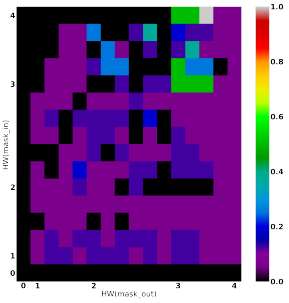
(a) SR on evolved_k S-Box with max of MI and noiseless measurements



(b) SR on evolved_k S-Box with min of MI and noiseless measurements



(c) SR on evolved_sr1 S-Box with max of MI and noiseless measurements



(d) SR on evolved_sr1 S-Box with min of MI and noiseless measurements

evolved_sr1/evolved_sr2/evolved_k S-Box

Table 4.7: Success Rate with the first most probable key guess

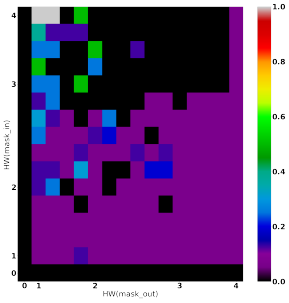
SNR	max MI	min MI
∞	✗/✗/✗	✓/✗/✓
10dB	✗/✗/✗	✗/✗/✗
1dB	✓/✓/✓	✗/✗/✗
-10dB	✓/✓/✓	✗/✗/✗

Table 4.8: Success Rate with the first two most probable key guesses

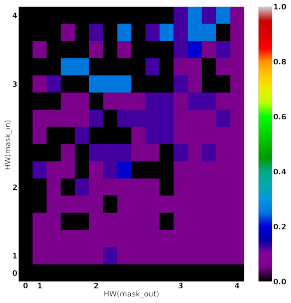
SNR	max MI	min MI
∞	✓/✓/✓	✓/✓/✓
10dB	✓/✓/✓	✓/✓/✗
1dB	✓/✓/✓	✗/✗/✗
-10dB	✓/✓/✓	✗/✗/✗

4.3.2. TA Resilient Substitution Boxes

There was only one substitution box that was generated for template attacks that was susceptible to MIA attacks that exploited the maximum of mutual information as a key distinguishing tool. On the other hand, the same substitution box offered a particular resistance when it was attacked with the minimum of mutual information:



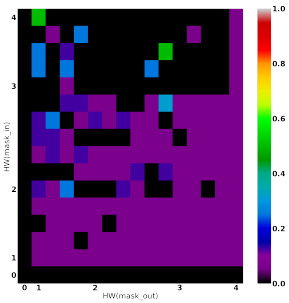
(a) SR on evolved_ta_sr2 S-Bo with max of MI and noiseless measurements



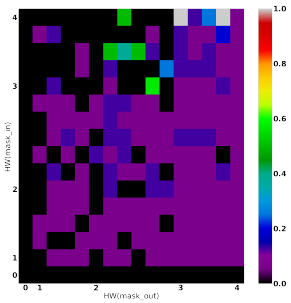
(b) SR on evolved_ta_sr2 S-Box with min of MI and noiseless measurements

The substitution box *evolved_ta_sr*4 had a behaviour similar to *evolved_to*, which means that it could not be found satisfying models for neither the minimum of mutual information nor the maximum of mutual information.

The substitution boxes *evolved_ta_sr*1 and *evolved_ta_sr*3 offered performance similar to *evolved_k* and *evolved_sr*1; however, for the substitution box *evolved_ta_sr*3 it was possible to identify two distinct leakage models that led to successful attack with the minimum of mutual information:



(a) SR on evolved_ta_sr1 S-Box with max of MI and noiseless measurements



(b) SR on evolved_ta_sr1 S-Box with min of MI and noiseless measurements

The second-order success rate would only point out some additional closest leakage models for 10dB SNR (and ∞SNR).

evolved_ta_sr1/evolved_ta_sr2/evolved_ta_sr3/evolved_ta_sr4 S-Box

Table 4.9: Success Rate with the first most probable key guess

SNR	max MI	min MI
∞	✗/✓/✗/✗	✓/✗/✓/✗
10dB	✗/✗/✗/✓	✓/✗/✗/✗
1dB	✓/✓/✓/✓	✗/✗/✗/✗
-10dB	✓/✓/✓/✓	✗/✗/✗/✗

Table 4.10: Success Rate with the first two most probable key guesses

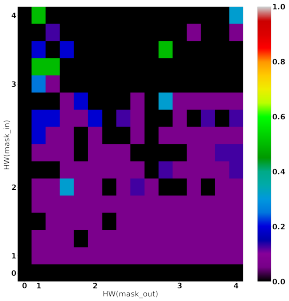
SNR	max MI	min MI
∞	✓/✓/✓/✓	✓/✗/✓/✗
10dB	✓/✓/✓/✓	✓/✗/✗/✗
1dB	✓/✓/✓/✓	✗/✗/✗/✗
-10dB	✓/✓/✓/✓	✗/✗/✗/✗

4.3.3. General Evaluation

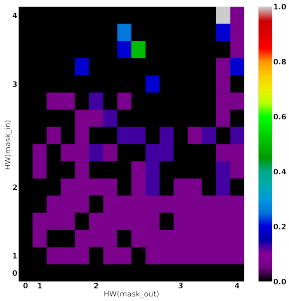
In line with the goals defined for this work, it could be identified as substitution boxes resilient to MIA attacks when it is used as distinguishing tool the maximum of mutual information the boxes: PRESENT, PRINCE, Klein, *evolved_cc*, *evolved_to*, *evolved_sr*2, *evolved_k*, *evolved_sr*1, *evolved_ta_sr*1, *evolved_ta_sr*3, and *evolved_ta_sr*4.

On such list, it could be identified a subset of substitution boxes that for a certain combination of variables $mask_{in}$ and $mask_{out}$ it is possible to retrieve a (sub-)key when it is used the minimum of mutual information as a distinguishing tool. The subset of such list is: PRINCE, Klein, *evolved_k*, *evolved_sr*1, *evolved_ta*$_s$*r*1, *evolved_ta_sr*3.

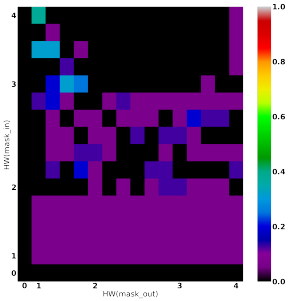
A subsequent investigation on a SNR profile of 10dB has shown that only for some of them (*evolved_ta_sr*1 and *Klein*), it was still be possible to find a leakage model for the retrieval of (sub-)keys with the minimum of mutual information:



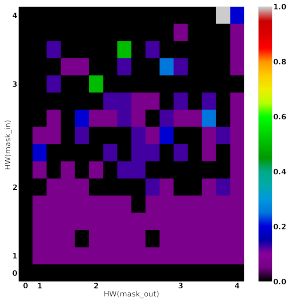
(a) SR on evolved_ta_sr1 S-Box with max of MI and SNR = 10dB



(b) SR on evolved_ta_sr1 S-Box with min of MI and SNR = 10dB



(c) SR on Klein's S-Box with max of MI and SNR = 10dB



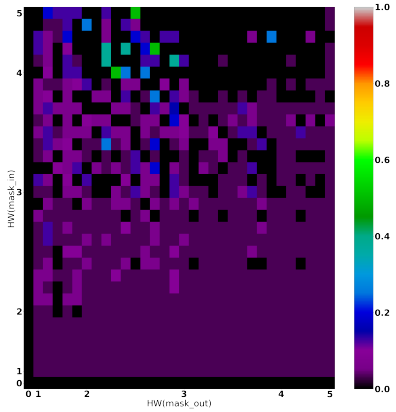
(d) SR on Klein's S-Box with min of MI and SNR = 10dB

For other SNR profiles (1dB and -10dB) it could not be possible to perform a successful side-channel attack to any substitution box with the minimum of mutual information as distinguishing tool. However, with the maximum of mutual information it was possible to identify leakage models that led to attacks with success rate equal to 1.

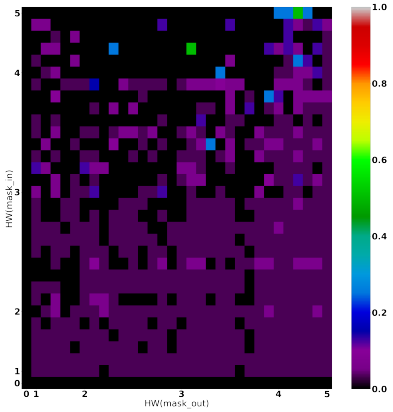
However, if it is taken into account also the second-order success rate, the substitution boxes that satisfy the first goal of this project are: evolved_cc and Klein. And for what concerns the second goal of the work, it was possible to find furthest leakage models only for Klein S-Box for SNR equal to ∞. It could be also found both furthest and closest leakage models for Klein for SNR equal to 10dB.

4.4. 5x5 Substitution Box

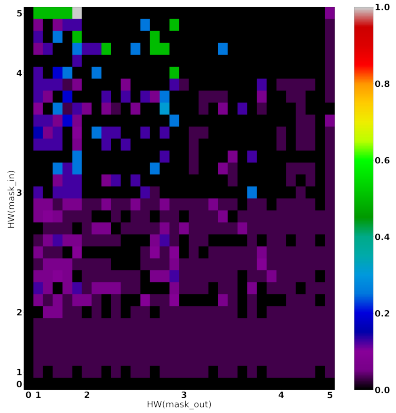
A first analysis of the substitution boxes ASCON and PRIMATE used as reference for the substitution boxes of Lerman et al.[16].



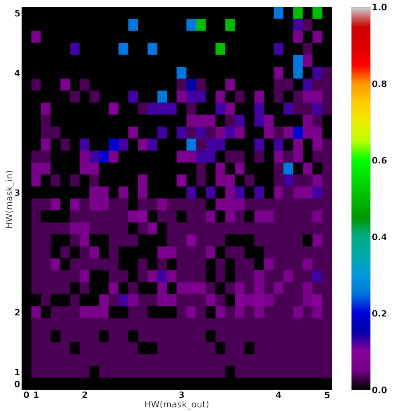
(a) SR of ASCON's S-Box with max of MI and noiseless measurements



(b) SR of ASCON's S-Box with min of MI and noiseless measurements



(c) SR on PRIMATE's S-Box with max of MI and noiseless measurements



(d) SR on PRIMATE's S-Box with min of MI and noiseless measurements

Where only for PRIMATE's S-Box was possible to find a leakage model that led to a successful attack with the maximum of mutual information. However, for neither ASCON's S-Box nor PRIMATE's S-Box was possible to find a leakage model that was exploitable when the minimum of mutual information was used as key distinguisher.

ASCON/PRIMATE S-Box

Table 4.11: Success Rate with the first most probable key guess

SNR	max MI	min MI
∞	✗/✓	✗/✗
10dB	✗/✓	✗/✗
1dB	✓/✓	✗/✗
-10dB	✓/✓	✗/✗

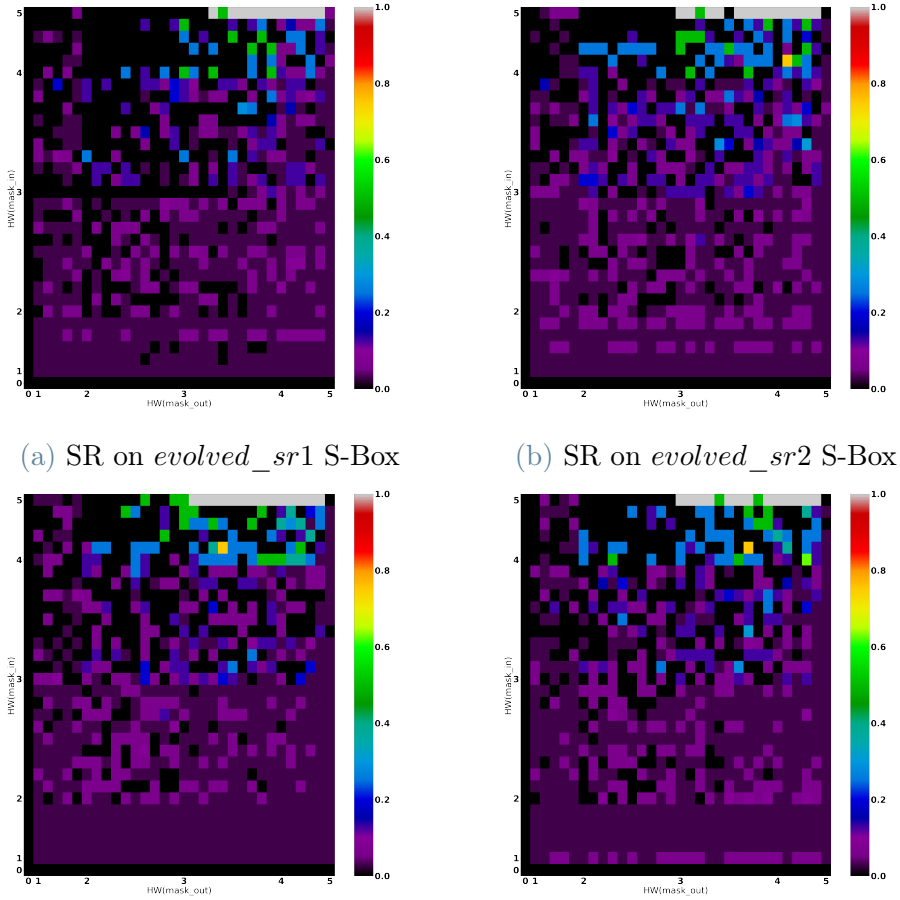
Table 4.12: Success Rate with the first two most probable key guesses

SNR	max MI	min MI
∞	✗/✓	✓/✓
10dB	✓/✓	✗/✗
1dB	✓/✓	✗/✗
-10dB	✓/✓	✗/✗

Second order success-rate would just point out furthest leakage models for SNR equal to ∞ and some closest leakage model for SNR equal to 10dB.

4.4.1. CPA Resilient Substitution Boxes

However, in the exploration of leakage models built from substitution boxes designed to be resilient to CPAs it was found out that for all the substitution boxes generated from genetic algorithm it was possible to find a particular leakage model where a (sub-)key was extracted with the minimum of mutual information with success rate equal to 1. In this case, all the substitution boxes that were generated offered better performances against CPA with respect to the ones used in real applications (like ASCON and PRIMATE):



(a) SR on *evolved_sr1* S-Box

(b) SR on *evolved_sr2* S-Box

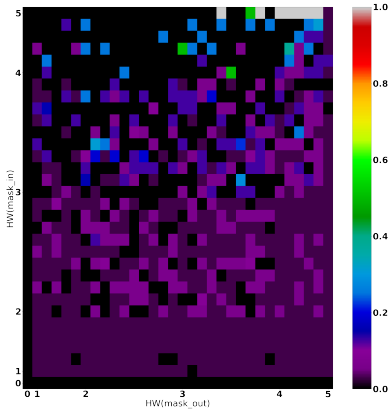
(c) SR on *evolved_sr5* S-Box

(d) SR on *evolved_sr6* S-Box

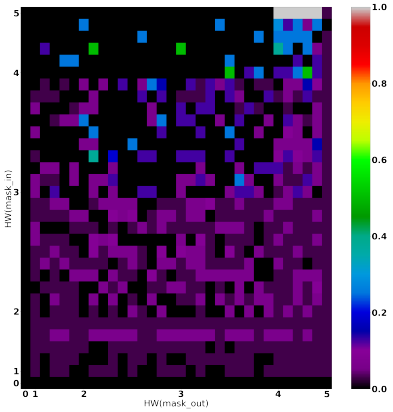
Figure 4.17: SR with noiseless measurements and min MI

Where it can be observed that it could be found a lot more furthest leakage models with respect to the substitution boxes analyzed previously.

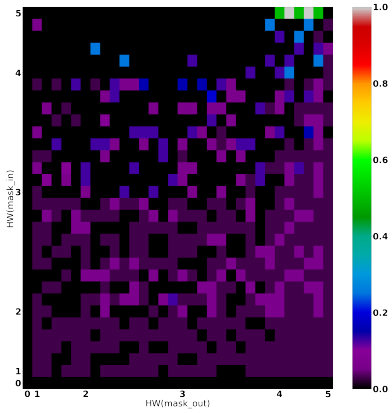
For the following substitution boxes it is still possible to exploit the minimum of mutual information, but with fewer leakage models.



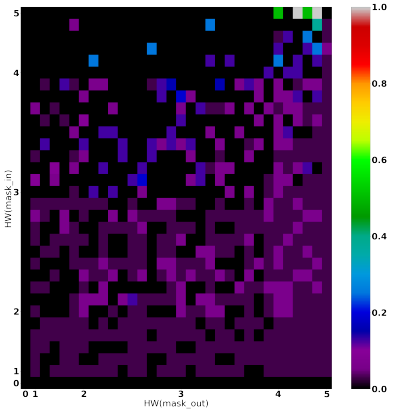
(a) SR on *evolved_sr3* S-Box



(b) SR on *evolved_sr4* S-Box



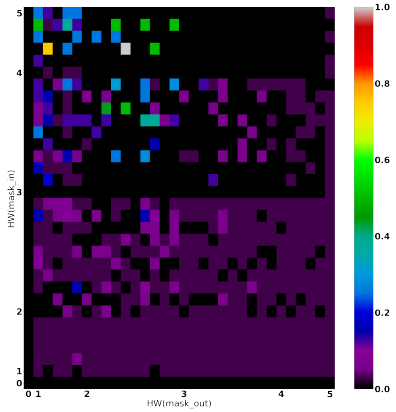
(c) SR on *evolved_sr8* S-Box



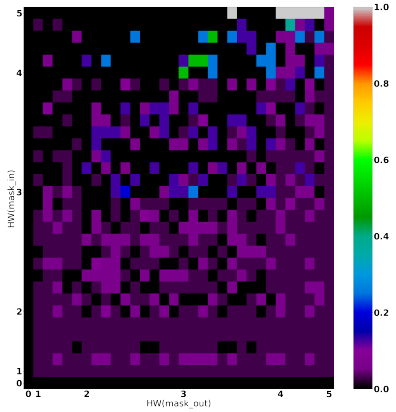
(d) SR on *evolved_sr9* S-Box

Figure 4.18: SR with noiseless measurements and min MI

And only for *evolved_sr7* it was also possible to find a closest leakage model where MIA was successful with success rate equal to 1.



(a) SR on *evolved_sr7* S-Box with max of MI and noiseless measurements



(b) SR on *evolved_sr7* S-Box with min of MI and noiseless measurements

evolved_sr1/evolved_sr2/evolved_sr3/evolved_sr4/evolved_sr5/evolved_sr6/evolved_sr7/evolved_sr8/evolved_sr9 S-Box

Table 4.13: Success Rate with the first most probable key guess

SNR	max MI	min MI
∞	✗/✗/✗/✗/✗/✗/✓/✗/✗	✓/✓/✓/✓/✓/✓/✓/✓/✓
10dB	✓/✗/✓/✓/✗/✗/✓/✗/✗	✓/✓/✓/✗/✓/✓/✗/✗/✗
1dB	✓/✓/✓/✓/✓/✓/✓/✗/✗	✗/✗/✗/✗/✗/✗/✗/✗/✗
-10dB	✓/✓/✗/✓/✗/✗/✓/✗/✗	✗/✗/✗/✗/✗/✗/✗/✗/✗

evolved_sr1/evolved_sr2/evolved_sr3/evolved_sr4/evolved_sr5/evolved_sr6/evolved_sr7/evolved_sr8/evolved_sr9 S-Box

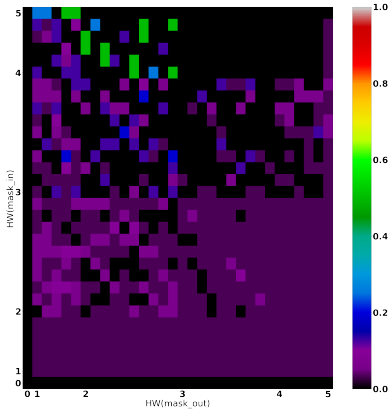
Table 4.14: Success Rate with the first two most probable key guesses

SNR	max MI	min MI
∞	✓/✗/✓/✓/✗/✗/✓/✗/✓	✓/✓/✓/✓/✓/✓/✓/✓/✓
10dB	✓/✗/✓/✓/✗/✗/✓/✓/✓	✓/✓/✓/✗/✓/✓/✗/✗/✗
1dB	✓/✓/✓/✓/✓/✓/✓/✓/✓	✗/✗/✗/✗/✗/✗/✗/✗/✗
-10dB	✓/✓/✓/✓/✗/✗/✓/✓/✓	✗/✗/✗/✗/✗/✗/✗/✗/✗

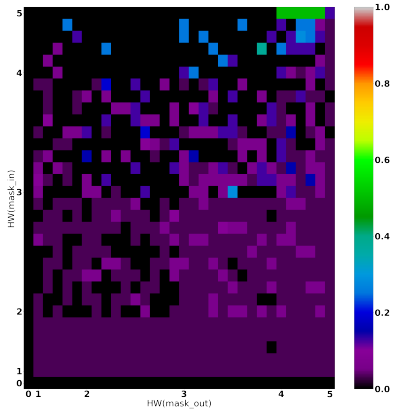
The second-order success rate would point out more exploitable closest leakage models for more substitution boxes. Unfortunately it would not help in finding additional furthest leakage models for those substitution boxes that did not have one (when SNR is 10dB).

4.4.2. TA Resilient Substitution Boxes

However, analysis of substitution boxes generated with genetic algorithm designed to be resilient to TAs was observed that only substitution box *evolved_ta_sr4* is resilient to both MIA attacks with maximum and minimum of mutual information:

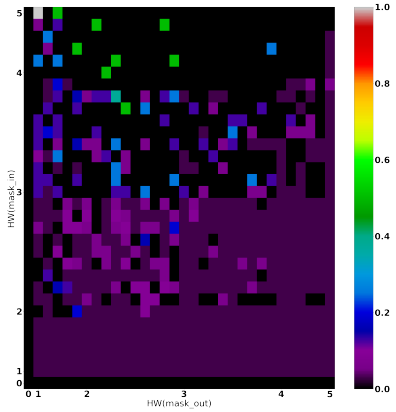


(a) SR on *evolved_ta_sr4* S-Box with max of MI and noiseless measurements

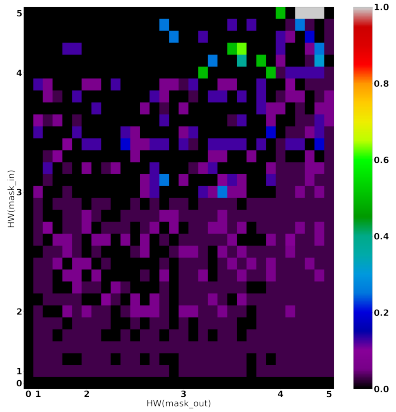


(b) SR on *evolved_ta_sr4* S-Box with min of MI and noiseless measurements

And the only substitution box for which it was possible to find leakage models that led to a successful attack with both maximum and minimum of mutual information is *evolved_ta_sr1*:

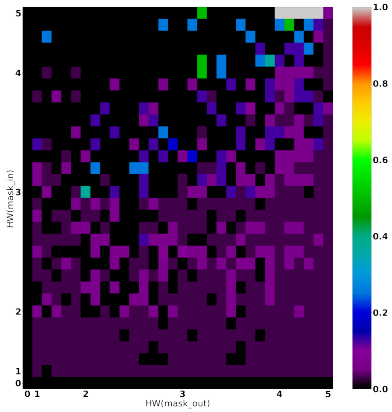


(a) SR on *evolved_ta_sr1* S-Box with max of MI and noiseless measurements

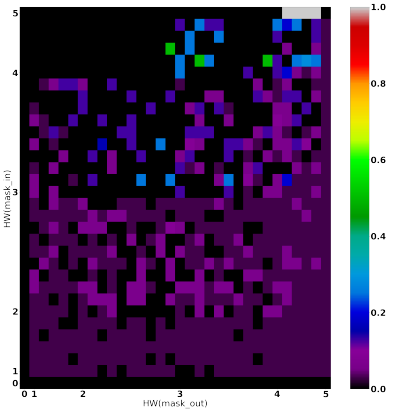


(b) SR on *evolved_ta_sr1* S-Box with min of MI and noiseless measurements

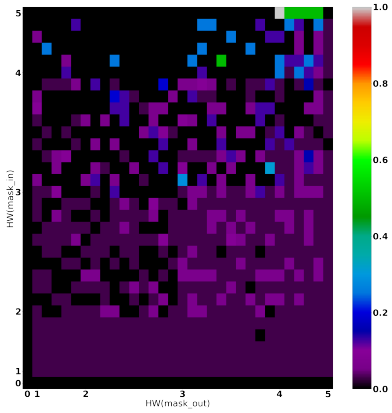
While for *evolved_ta_sr6*, *evolved_ta_sr5*, *evolved_ta_sr3* and *evolved_ta_sr2* S-Boxes it was possible to identify leakage models exploitable only with the minimum of mutual information:



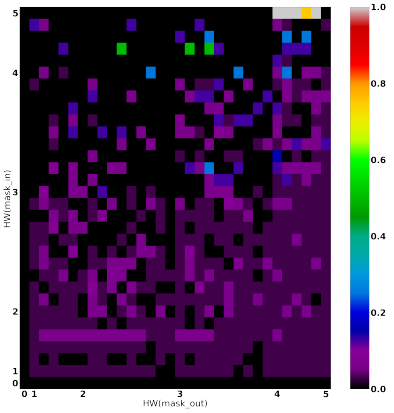
(a) SR on *evolved_ta_sr6* S-Box



(b) SR on *evolved_ta_sr5* S-Box



(c) SR on *evolved_ta_sr3* S-Box



(d) SR on *evolved_ta_sr2* S-Box

Figure 4.22: SR with noiseless measurements and min MI

However, for any of the substitution boxes resilient to TAs it was possible to find leakage models where the minimum of mutual information would lead to a successful attack with a satisfying success rate when side-channel measurements had a SNR of 10dB (or smaller).

evolved_ta_sr1/evolved_ta_sr2/evolved_ta_sr3/evolved_ta_sr4/ evolved_ta_sr5/evolved_ta_sr6 S-Box

Table 4.15: Success Rate with the first most probable key guess

SNR	max MI	min MI
∞	✓/✗/✗/✗/✗/✗	✓/✓/✓/✗/✓/✓
10dB	✓/✓/✗/✗/✓/✓	✗/✗/✗/✗/✗/✗
1dB	✓/✓/✓/✓/✓/✓	✗/✗/✗/✗/✗/✗
-10dB	✓/✓/✓/✓/✓/✓	✗/✗/✗/✗/✗/✗

evolved_ta_sr1/evolved_ta_sr2/evolved_ta_sr3/evolved_ta_sr4/ evolved_ta_sr5/evolved_ta_sr6 S-Box

Table 4.16: Success Rate with the first two most probable key guesses

SNR	max MI	min MI
∞	✓/✓/✗/✗/✗/✓	✓/✓/✓/✓/✓/✓
10dB	✓/✓/✓/✓/✓/✓	✗/✗/✗/✗/✗/✗
1dB	✓/✓/✓/✓/✓/✓	✗/✗/✗/✗/✗/✗
-10dB	✓/✓/✓/✓/✓/✓	✗/✗/✗/✗/✗/✗

Second-order success rate would only help into finding some furthest leakage models for *evolved_ta_sr*4 substitution box; it would also help finding some closest leakage model for SNR equal to 10dB.

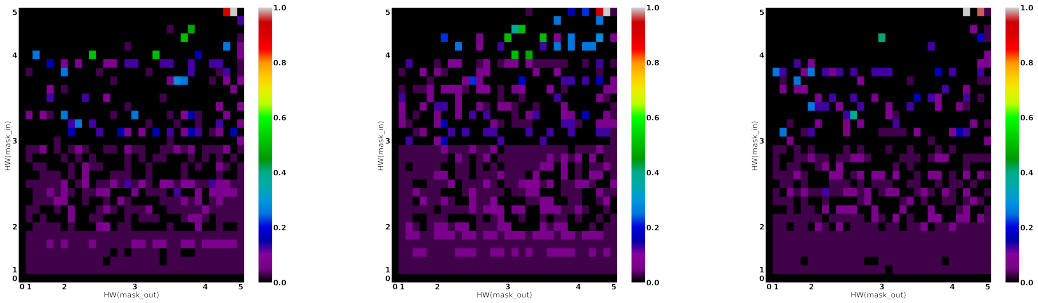
4.4.3. General Evaluation

The set of substitution boxes that satisfied the first goal of this project is made of: ASCON's S-Box, *evolved_sr*1, *evolved_sr*2, *evolved_sr*3, *evolved_sr*4, *evolved_sr*5, *evolved_sr*6, *evolved_sr*8, *evolved_sr*9, *evolved_ta_sr*2, *evolved_ta_sr*3, *evolved_ta_sr*4, *evolved_ta_sr*5, and *evolved_ta_sr*6.

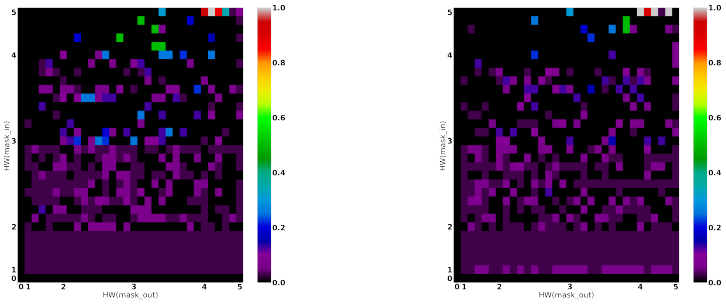
From the list of S-Boxes cited above, the substitution boxes where it was possible to build leakage models that lead to successful MIA attacks with the minimum of mutual

information are: *evolved_sr1*, *evolved_sr2*, *evolved_sr3*, *evolved_sr4*, *evolved_sr5*, *evolved_sr6*, *evolved_sr8*, *evolved_sr9*, *evolved_ta_sr2*, *evolved_ta_sr3*, *evolved_ta_sr5*, and *evolved_ta_sr6*.

An analysis on those models with measurements having a SNR of 10dB has shown that only for CPA resilient substitution boxes was still possible to build leakage models that lead to successful attacks with the minimum of mutual information and success rate equal to 1:



(a) SR on *evolved_sr1* S-Box (b) SR on *evolved_sr2* S-Box (c) SR on *evolved_sr3* S-Box



(d) SR on *evolved_sr5* S-Box (e) SR on *evolved_sr6* S-Box

Figure 4.23: SR with measurements with SNR = 10dB and min MI

If it is taken into account the second-order success rate, the substitution boxes satisfying the first goal of the work are: ASCON's substitution box, *evolved_sr2*, *evolved_sr5*, *evolved_sr6*, *evolved_sr8*, *evolved_ta_sr2*, *evolved_ta_sr3*, *evolved_ta_sr5*. Of which, the substitution boxes that also satisfies the second goal are the same: ASCON's S-Box, *evolved_sr2*, *evolved_sr5*, *evolved_sr6*, *evolved_sr8*, *evolved_ta_sr2*, *evolved_ta_sr3*, *evolved_ta_sr5*. However, only for *evolved_sr2*, *evolved_sr5*, *evolved_sr6* it was also possible to find furthest leakage models for SNR equal to 10dB.

5 | Conclusions and Future Developments

In this work it has been analyzed several intermediate functions that are mainly used to build hypothetical leakage models for generic side-channel attacks.

These leakage models were obtained by exhaustive search of variables $mask_{in}$ and $mask_{out}$ of hypothetical leakage models $\hat{L}_{k_{hyp}} := mask_{out}$ & $S\text{-}Box((m$ & $mask_{in}) \oplus k_{hyp})$.

It was first analyzed the success rate with respect to the maximum of mutual information so as to identify such intermediate functions that offered the highest resilience possible against side-channel attacks performed with mutual information analysis.

Then, on such intermediate functions, it was analyzed if it could be found leakage models that would systematically perform the worst possible only when interrogated with the correct secret key. Then, these models, if they do exist, they would reveal the secret key in correspondence with the minimum of mutual information.

In this project it was possible to identify such intermediate function, for both 4-to-4 bits functions and 5-to-5 bits functions introduced by Lerman et al.[16]. These substitution boxes highlight a parallel attack surface that can be used in conjunction with closest leakage models that exploits the minimum of mutual information as a key distinguishing tool. These furthest leakage models can be used in conjunction to the closest leakage models in order to mount successful side-channel attacks.

It is also worth noticing that it was possible to identify the majority of furthest leakage models on those substitution boxes that were designed to withstand CPA attacks.

However, such models were only observed when side-channel measurements were simulated with high SNR, that can be justified only from a very motivated actor, capable to capture high quality side-channel measurements with a SNR of at least 10dB.

In spite of having requirements that are tough to be met by an average adversary, these furthest models do still exist, thus they can be used to perform successful generic SCAs.

A future work might involve a further analysis of metrics used for the categorization of side-channel attacks, e.g. by means of the *Guessing Entropy*. In fact, guessing entropy might help giving more information on the average position of guessed keys for interesting leakage models identified with the success rate metric used in this project.

Other works may also investigate if it could be still possible to extract the secret key by searching the minimum of mutual information when countermeasures of different nature are implemented on a target device. This would also verify whether the minimum of mutual information would still work for the multivariate mutual information analysis.

In addition to it, it could be also explored if it may be required on average more or less side-channel measurements when MIA attacks are performed with furthest leakage models with respect to closest leakage models.

Bibliography

[1] Abramowitz and Stegun. *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*. United States Department of Commerce, National Bureau of Standards (NBS), 1964.

[2] Lejla Batina et al. "Mutual Information Analysis: A Comprehensive Study". *J. Cryptol.* 24.2 (Apr. 2011), pp. 269–291. ISSN: 0933-2790. DOI: 10.1007/s00145-010-9084-8. URL: https://doi.org/10.1007/s00145-010-9084-8.

[3] Paul Bottinelli and Joppe W. Bos. "Computational aspects of correlation power analysis". *Journal Of Cryptographic Engineering* 7.3 (2017), pp. 15. 167–181. DOI: 10.1007/s13389-016-0122-9. URL: http://infoscience.epfl.ch/record/232881.

[4] Eric Brier, Christophe Clavier, and Francis Olivier. "Correlation Power Analysis with a Leakage Model". *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*. Vol. 3156. Lecture Notes in Computer Science. Springer, 2004, pp. 16–29. DOI: 10.1007/978-3-540-28632-5_2. URL: https://iacr.org/archive/ches2004/31560016/31560016.pdf.

[5] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. "Template Attacks". *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. Vol. 2523. Lecture Notes in Computer Science. Springer, 2002, pp. 13–28. DOI: 10.1007/3-540-36400-5_3.

[6] Suresh Chari et al. "Towards Sound Approaches to Counteract Power-Analysis Attacks". *Advances in Cryptology — CRYPTO 99*. Ed. by Michael Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 398–412.

[7] Éloi Chérisey et al. "On the Optimality and Practicability of Mutual Information Analysis in Some Scenarios". *Cryptography Commun.* 10.1 (Jan. 2018), pp. 101–121. ISSN: 1936-2447. DOI: 10.1007/s12095-017-0241-x. URL: https://doi.org/10.1007/s12095-017-0241-x.

[8] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. "Differential Power Analysis in the Presence of Hardware Countermeasures". *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*. Vol. 1965. Lecture Notes in Computer Science. Springer, 2000, pp. 252–263. DOI: 10.1007/3-540-44499-8_20.

[9] D. Freedman and P. Diaconis. Vol. 57(4). Springer, 1981, pp. 453–476.

[10] Benedikt Gierlichs et al. "Mutual Information Analysis". *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*. Vol. 5154. Lecture Notes in Computer Science. Springer, 2008, pp. 426–442. DOI: 10.1007/978-3-540-85053-3_27. URL: https://www.iacr.org/archive/ches2008/51540423/51540423.pdf.

[11] Ilham Hassoune et al. "Dynamic Differential Self-Timed Logic Families for Robust and Low-Power Security ICs". *Integr. VLSI J.* 40.3 (2007), pp. 355–364. ISSN: 0167-0260.

[12] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. *Good is Not Good Enough: Deriving Optimal Distinguishers from Communication Theory*. Cryptology ePrint Archive, Report 2014/527. https://ia.cr/2014/527. 2014.

[13] Marc Joye, Pascal Paillier, and Berry Schoenmakers. "On Second-Order Differential Power Analysis". Aug. 2005, pp. 293–308.

[14] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25.

[15] Liran Lerman et al. "Higher order side-channel attack resilient S-boxes". English. *2018 ACM International Conference on Computing Frontiers, CF 2018 - Proceedings*. Ed. by D.R. Kaeli and M. Pericàs. 15th ACM International Conference on Computing Frontiers, CF 2018 ; Conference date: 08-05-2018 Through 10-05-2018. United States: Association for Computing Machinery (ACM), 2018, pp. 336–341. ISBN: 978-1-4503-5761-6. DOI: 10.1145/3203217.3206428.

[16] Liran Lerman et al. "On the Construction of Side-Channel Attack Resilient S-boxes". *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*. Ed. by Sylvain Guilley. Vol. 10348. Lecture Notes in Computer Science. Springer, 2017, pp. 102–119. DOI: 10.1007/978-3-319-64647-3_7. URL: https://doi.org/10.1007/978-3-319-64647-3%5C_7.

[17] Vcitor Lomné et al. "How to Estimate the Success Rate of Higher-Order Side-Channel Attacks". *Proceedings of the 16th International Workshop on Cryptographic Hardware and Embedded Systems — CHES 2014 - Volume 8731*. Berlin, Heidelberg: Springer-Verlag, 2014, pp. 35–54. ISBN: 9783662447086.

[18] Stefan Mangard. "Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness". *Topics in Cryptology – CT-RSA 2004*. Ed. by Tatsuaki Okamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 222–235.

[19] J.L. Massey. "Guessing and entropy". *Proceedings of 1994 IEEE International Symposium on Information Theory*. 1994, pp. 204–. DOI: 10.1109/ISIT.1994.394764.

[20] Thomas S. Messerges. "Using Second-Order Power Analysis to Attack DPA Resistant Software". *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*. Vol. 1965. Lecture Notes in Computer Science. Springer, 2000, pp. 238–251.

[21] Amir Moradi et al. "A Comparative Study of Mutual Information Analysis under a Gaussian Assumption". Jan. 2009, pp. 193–205. ISBN: 978-3-642-10837-2. DOI: 10.1007/978-3-642-10838-9_15.

[22] Amir Moradi et al. "A Comparative Study of Mutual Information Analysis under a Gaussian Assumption". *Information Security Applications: 10th International Workshop, WISA 2009, Busan, Korea, August 25-27, 2009, Revised Selected Papers*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 193–205.

[23] E. Oswald et al. "Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers." *Topics in Cryptology - CT-RSA 2006, The Cryptographers Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*. Ed. by D. Pointcheval. Vol. 3860. Lecture Notes in Computer Science. Springer, 2006, pp. 192–207.

[24] Mark Randolph and William Diehl. "Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman". *Cryptography* 4.2 (2020). ISSN: 2410-387X. DOI: 10.3390/cryptography4020015. URL: https://www.mdpi.com/2410-387X/4/2/15.

[25] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. "Generic DPA Attacks: Curse or Blessing?" Apr. 2014, pp. 98–111. ISBN: 978-3-319-10174-3. DOI: 10.1007/978-3-319-10175-0_8.

[26] Abul Sarwar. "CMOS Power Consumption and Cpd Calculation" (1997).

[27] D.W. Scott. Vol. 66(3). Biometrika Trust, 1979, pp. 605–610.

[28] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks". *Advances in Cryptology*

- *EUROCRYPT 2009*. Ed. by Antoine Joux. Berlin, Heidelberg: Springer Berlin Hfoneidelberg, 2009, pp. 443–461. ISBN: 978-3-642-01001-9.

[29] Steve Summit. *comp.lang.c FAQ list - Question 13.20*. visited on 25-02-2022. URL: http://c-faq.com/lib/gaussian.html.

[30] Stefan Tillich, Christoph Herbst, and Stefan Mangard. "Protecting AES Software Implementations on 32-Bit Processors Against Power Analysis". Vol. 4521. Jan. 2007, pp. 141–157. ISBN: 978-3-540-72737-8. DOI: 10.1007/978-3-540-72738-5_10.

[31] N. Veyrat-Charvillon and sF.-X. Standaert. "Generic side-channel distinguishers: Improvements and limitations - CRYPTO 2011". Ed. by P. Rogaway. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 354–372.

[32] Jason Waddle and David A. Wagner. "Towards Efficient Second-Order Power Analysis". *CHES*. 2004.

[33] Carolyn Whitnall. "An information theoretic assessment of first-order mia". 2010.

List of Figures

1.1 AES encryption structure 5
1.2 AES decryption structure 6
1.3 DES encryption and decryption structures 8

2.1 SPA on a DES encryption 17
2.2 SPA on a AES encryption 18
2.3 SPA on a round computation of AES' cipher 18
2.4 Portion of a power trace of a computation of a RSA private key 20
2.5 DPA attack, with one successful attack and two unsuccessful attack 24
2.6 Correlation values for CPA 29
2.7 Mutual information values of MIA 34
2.8 Histogram and Kernel method respectively 37

4.1 SR on AES's S-Box with noiseless measurements 60
4.2 SR on AES's S-Box with min MI and 10dB SNR measurements 61
4.7 SR on DES, ideal measurements, CPA 64
4.8 SR with noiseless measurements and max MI 65
4.9 SR with noiseless measurements and min MI 66
4.17 SR with noiseless measurements and min MI 75
4.18 SR with noiseless measurements and min MI 76
4.22 SR with noiseless measurements and min MI 80
4.23 SR with measurements with SNR = 10dB and min MI 82

List of Tables

4.1 Success Rate with the first most probable key guess 63
4.2 Success Rate with the first most probable key guess 65
4.3 Success Rate with the first most probable key guess 66
4.4 Success Rate with the first two most probable key guesses 66
4.5 Success Rate with the first most probable key guess 68
4.6 Success Rate with the first two most probable key guesses 68
4.7 Success Rate with the first most probable key guess 69
4.8 Success Rate with the first two most probable key guesses 69
4.9 Success Rate with the first most probable key guess 71
4.10 Success Rate with the first two most probable key guesses 71
4.11 Success Rate with the first most probable key guess 74
4.12 Success Rate with the first two most probable key guesses 74
4.13 Success Rate with the first most probable key guess 77
4.14 Success Rate with the first two most probable key guesses 77
4.15 Success Rate with the first most probable key guess 81
4.16 Success Rate with the first two most probable key guesses 81

Ringraziamenti

Vorrei innanzitutto ringraziare di cuore i miei relatori il Prof. Alessandro Barenghi ed il Prof. Gerardo Pelosi per avermi guidato con pazienza e disponibilità lungo questo percorso impegnativo. La vostra esperienza è stata per me essenziale ed intestimabile, e per questo ve ne sono immensamente grato.

Mi sento anche di ringraziare tutte le persone in cui mi sono imbattuto durante l'intera carriera universitaria. Sia direttamente che indirettamente, che si sia sviluppata o meno una relazione, la vostra influenza ha fatto sì che intraprendessi questo percorso, e per questo vi sono riconoscente.

Infine, ma non per importanza, ringrazio i miei genitori per avermi sempre sostenuto, anche nei momenti più difficili. Vi ringrazio dal profondo del cuore per avermi dato la possibilità di intrapendere e concludere questo fantastico viaggio.