University of Piraeus – Department of Informatics

Postgraduate Studies Program

Advanced Computer Systems

Postgraduate Dissertation

Τίτλος	«ΕΠΙΘΕΣΗ ΜΕ ΑΝΑΛΥΣΗ ΣΥΣΧΕΤΙΣΗΣ ΙΣΧΥΟΣ ΣΕ ΥΛΟΠΟΙΗΣΗ ΤΟΥ AES ΜΕ ΕΝΣΩΜΑΤΩΜΕΝΟ ΛΟΓΙΣΜΙΚΟ», «CORRELATION POWER ANALYSIS ATTACK ON EMBEDDED SOFTWARE IMPLEMENTATION OF AES»
Όνομα Επίθετο	ΝΟΜΙΚΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
Πατρώνυμο	ΝΙΚΟΛΑΟΣ
Αριθμός Μητρώου	ΜΠΣΠ17052
Επιβλέπων	ΨΑΡΑΚΗΣ ΜΙΧΑΛΗΣ ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

Date	25/07/2019

Τριμελής Εξεταστική Επιτροπή

Ψαράκης Μιχάλης Επίκουρος Καθηγητής	Κοτζανικολάου Παναγιώτης Επίκουρος Καθηγητής	Πατσάκης Κων/νος Επίκουρος Καθηγητής

Abstract

Implementations of cryptographic algorithms often leak information through various means, such as power consumption, electromagnetic emission, execution time, heat or even sound. Every attack that aims on taking advantage of these kinds of leakage is called Side Channel Attack (SCA). Power analysis (PA) SCA attacks focus on studying power consumption during the execution of an algorithm. In this master thesis we study PA SCA against software level implementations of AES-128 running at a development board which integrates an ARM®32-bit Cortex®-M3 microprocessor. Measuring and analyzing the power consumption of our development board can reveal the secret key of the encryption. Thus, there is a major security risk which is necessary to be considered at the implementation level. Countermeasures against SCA can be implemented both at hardware and software level. One of the most powerful countermeasures against PA SCA is masking. In masking technique, we randomize the intermediate values of an algorithm by concealing it with a random value (mask). We perform attacks both on unprotected and masked software implementations of AES-128. We manage to acquire the secret key of an unprotected AES-128 implementation and present the security of masking technique against first order attacks.

Περίληψη

Οι υλοποιήσεις κρυπτογραφικών αλγορίθμων συχνά κατά την εκτέλεση τους, διαρρέουν πληροφορία μέσω διαφόρων μέσων, όπως η κατανάλωση ενέργειας, η ηλεκτρομαγνητική εκπομπή, ο χρόνος εκτέλεσης, η θερμότητα ή ακόμα και ο ήχος. Κάθε επίθεση που έχει ως στόχο την εκμετάλλευση αυτών των ειδών διαρροής ονομάζεται Επίθεση Πλευρικού Καναλιού (ΕΠΚ). Οι επιθέσεις ανάλυσης ισχύος (AI) ΕΠΚ επικεντρώνονται στη μελέτη της κατανάλωσης ενέργειας κατά την εκτέλεση ενός αλγορίθμου. Σε αυτή τη μεταπτυχιακή εργασία μελετάμε τις ΕΠΚ AI ενάντια στην υλοποίηση του AES-128 σε επίπεδο λογισμικού που υλοποιείται σε μια πλατφόρμα ανάπτυξης που ενσωματώνει έναν μικροεπεξεργαστή ARM®32-bit Cortex®-M3. Η μέτρηση και η ανάλυση της κατανάλωσης ρεύματος της πλατφόρμας ανάπτυξης μπορεί να αποκαλύψει το μυστικό κλειδί της κρυπτογράφησης. Επομένως, υπάρχει ένας σημαντικός κίνδυνος ασφάλειας, ο οποίος είναι απαραίτητο να εξεταστεί σε επίπεδο εφαρμογής. Τα αντίμετρα κατά των ΕΠΚ μπορούν να εφαρμοστούν τόσο σε επίπεδο υλικού όσο και λογισμικού. Ένα από τα πιο ισχυρά αντίμετρα εναντίον του ΕΠΚ AI είναι το masking. Στην τεχνική του masking, τυχαιοποιούμε τις ενδιάμεσες τιμές ενός αλγορίθμου, αποκρύπτοντάς τις με τυχαίες τιμές (μάσκες). Πραγματοποιούμε επιθέσεις τόσο σε μη προστατευμένες όσο και σε προστατευμένες υλοποιήσεις λογισμικού του AES-128. Καταφέρουμε να αποκτήσουμε το μυστικό κλειδί μιας απροστάτευτης υλοποίησης του AES-128 και να παρουσιάσουμε την ασφάλεια της τεχνικής masking κατά των επιθέσεων πρώτης τάξης.

Table of Contents

1 Introduction

Information technology is evolving at an amazing pace. Personal computers, fax machines, pagers, and mobile phones are being used by millions of people worldwide. Similarly, the interest on smart card technology has soared since the 1990's, and up until now the number and variety of smart card-based applications exploded around the world. In an increasing number of large and important smart card-based systems, from pay-TV through GSM mobile phones and prepaid gas meters to electronic wallets, smart cards are used by millions of cardholders worldwide in more than 90 countries, primarily in Europe and the Far East, processing point-of-sale transactions, managing records, and protecting computers and secure facilities. For these reasons information engineers have developed an increasing interest in the tamper resistance properties of smart cards and other special purpose security processors. Tamper resistance is not absolute: an attacker with access to semiconductor test equipment can retrieve secret keys from a smart card controller by direct observation and manipulation of the chip's components.

It is generally believed that, given sufficient investment, any chip-sized tamper resistant device can be penetrated in this way. This is the reason for the increasing interest in new attack methods on one side and in new countermeasures and new cryptographic algorithms on the other hand. Cryptographic algorithms are building blocks of many security protocols and can be implemented both in software and hardware. Software solutions are cheaper and more flexible, while hardware implementations provide higher speed and intrinsic security. A trade-off in cost and speed can be achieved by hardware-software co-design.

On the other hand, the security of each implementation also needs to be considered. Namely, attacks on cryptographic algorithms are usually divided into mathematical and hardware attacks. The latter are based on lacking of protection against of an implementation against passive or active hardware attacks. Passive attacks benefit from side channel information, which is collected by measuring some physical quantity. More precisely, while secret data are being processed, they can be deduced by observing execution time, power consumption, electromagnetic radiation, etc. The second class of hardware attacks, i.e. the active attacks, is more invasive as they are based on the introduction of faults, which result in erroneous calculations leading to the exposure of the secret key using techniques such as Differential Fault analysis (DFA) [[1]]. The usual cause of these faults can be sudden changes, i.e. glitches, in various parameters such as power supply, clock, temperature, etc. An attacker could also use a light flash with equipment such as a camera flash or a laser in order to induce a fault [[2], [3]].

Cryptographic devices have several physical and logical interfaces and therefore a second criterion for categorizing an attack on a cryptographic device is the interface that is exploited by the attack. It is possible to distinguish between invasive and non-invasive attacks. An invasive attack is the strongest type of attack that can be mounted on a cryptographic device. An invasive attack typically starts with the de-packaging of the device. Subsequently, different components of the device are accessed directly using a probing station. This part of an invasive attack can be passive if the probing station is only used to observe data signals or active if signals in the device are changed to alter the functionality of the device. In a non-invasive attack, the cryptographic device is essentially attacked as it is and only directly accessible interfaces are exploited. Most non-invasive attacks can be conducted with relatively inexpensive equipment, and hence, these attacks pose a serious practical threat to the security of cryptographic devices.

Passive non-invasive attacks are often also referred to as side-channel attacks. Side-channel attacks can be also active non-invasive attacks. The goal of these attacks is to insert a fault (for example clock or power glitches) in a device without depackaging it. Regarding attacks, this thesis focuses exclusively on power analysis attacks even if a general overview also of the other types of attacks is treated in the following.

Data and computations are the main variables of a physical structure, for example, the charge on a capacitor or a transistor's state. As embedded Systems increasingly find applications in communication, medical object, tracking, and other services an adversary can access a device and analyze the device's physical quantities. Leaked physical information can be used to extract secret data by side-channel attacks. Side channel attacks (SCA) concern all the possible attacks targeting the physical implementation of an algorithm. Regarding cryptographic algorithms strong effort takes place in proving a high level of theoretical security. SCA aims on making use of the information leakage of an implementation to break this theoretical security [[4]].

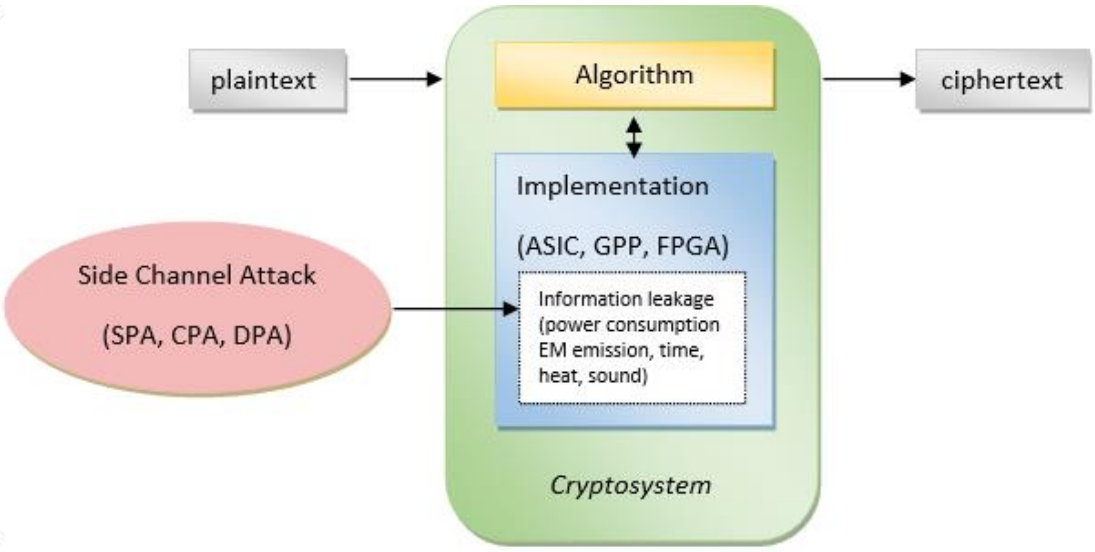


Figure 1 example crypto

We define as a cryptosystem as shown in figure1 an algorithm that takes an input and through specific operations produces a ciphertext. The algorithm itself is secure but the physical implementation could leak information. This information leakage can relate to power consumption, electromagnetic emission, execution time, heat or sound. SCA aims at exploiting information about the algorithm through this leakage.

Nowadays the Internet of things (IoT) is the present and the future of the technological changes in every person life. IoT extends the connectivity of physical devices and creates automatic systems that can work without the direct need of human interaction. Emerging IoT fields are embedded systems, wireless sensor networks, control systems, automation (home and building automation), real-time analytics, machine learning. Almost every aspect of IoT interacts with data which privacy and security must be guaranteed. Most of the times, regarding embedded systems, one main goal of IoT is low power consumption in order to achieve greater power autonomy. This fact usually means that IoT embedded systems use CPUs with low processing power. Attacking such systems is nowadays technically and financially (concerning the cost of the equipment needed) feasible.

Considering the systematic classification of SCAs shown in [[5]] we can clearly realize the high level of security risk that SCA can pose nowadays. We will present parameters of electronic devices which increase dramatically the level of likelihood of a SCA attack to be able to leak the secret key of a device.

- always-on and portability: These factors increase the availability of a targeted device. This fact means that the adversary has more time and more chances of studying and attacking a device.
- bring your own device (BYOD): Accessing critical infrastructures with personal devices increases the risk of creating a security breach through the personal connected devices.

- ease of software installation: Most devices have simple ways installing different software. Because of this, malicious software can easily penetrate in a device.
- OS based on Linux kernel: Most systems based on Linux kernel were manufactured to be implemented on desktop computers systems. Implementing them on mobile embedded devices can cause lot of security issues.
- features and sensors,

There are various types of SCA. The classification is usually based on the method that each type of attack uses to acquire the information leakage. SCA attacks have been an active research field for nearly two decades, a class of which focuses on power analysis (PA). PA attacks are being considered a serious threat which has to be considered by the designers of cryptographic algorithms. The attack makes use of the power consumption of a cryptographic implementation and is able to reveal information about the data being processed. The first power analysis attacks were conducted by P. Kocher et al. [[6]] using Differential Power Analysis. In [5], the authors present various power analysis attacks against smart cards.

In order to conduct a PA attack, we should first measure the power consumption of the targeted device. The measurement of the power consumption through time is called a power trace. A simple example of a power trace can be seen in figure 2. PA attacks manage to extract information for the data being processed by analyzing these power traces. Thus, power consumption leakage leads to security information leakage.

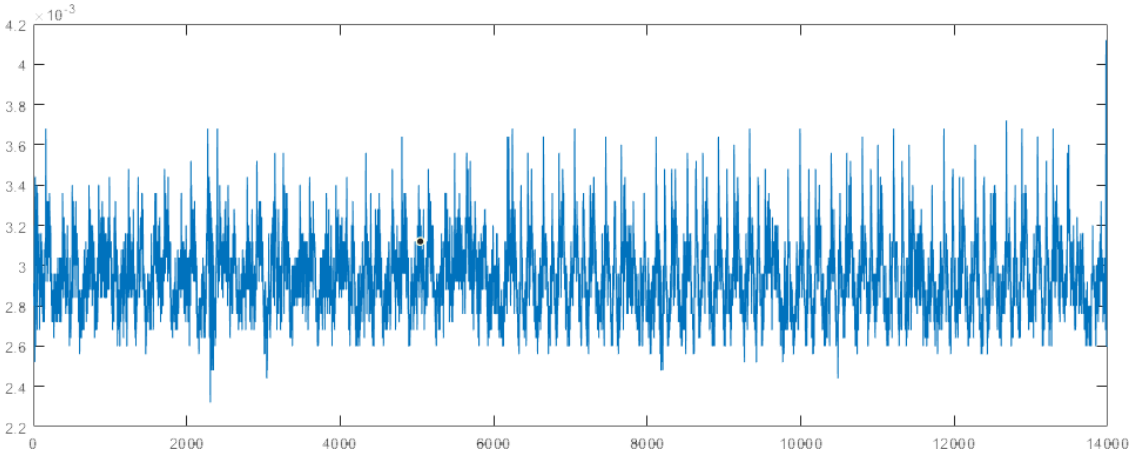


Figure 2 Typical power trace. Axis x presents voltage in millivolt and axis y presents time in time points.

PA attacks can be separated into two main categories, simple power analysis (SPA) and differential power analysis (DPA). In SPA attacks the attacker tries to acquire information directly from a given trace. This method requires detailed information about the implementation of the algorithm which is being attacked[[7]]. On the other hand, DPA attacks [6] are more effective because they can be used even if detailed information about the algorithm and its implementation are not available and can reveal information even if the recorded traces contain more noise. DPA attacks take advantage of how the power consumption at fixed moments of time depends on the processed data using Correlation Power Analysis [[8]] in order to obtain the secret key of the encryption.

Given the existing vulnerability of embedded devices implementing cryptographic algorithms against power analysis attacks many countermeasures have been proposed. The main goal of every countermeasure is to reduce the leakage of secret information in the power consumption of a device or to conceal it using the addition of noise. The two main categories in which we could categorize the countermeasures are hiding and masking.

Hiding is a countermeasure which focuses on either adding noise which can hide the information being leaked or by attempting to remove the data dependency from the power traces . Hiding can be achieved by various methods such as time-based implementations with random insertion of dummy operations,

shuffling and amplitude-based implementations such as targeted selection of instructions (not every instruction leaks the same amount of power), specific usage of memory addresses or parallel activity.

Masking is a countermeasure technique which focuses on randomization of intermediate values of a cryptographic computation [[9]]. Masking can be either Boolean or arithmetic. A masked intermediate value Vm is an intermediate value v which is concealed by a random value m: Vm = V * m. The power consumption which is being measured during the execution of the cryptographic algorithm corresponds to the computation of the masked values. In general masking is based on a secret-sharing scheme that uses two shares V and m. If we apply n masks on the same intermediate value, we can prevent a n-th order DPA Attack. At the end of every masking technique the same masks are being applied on the masked data so that the final ciphertext is the same as if there was no masking applied on AES algorithm.

The secure communication between two or more entities is the main goal of a cryptosystem. Usually the encryption and decryption of an input is based on a secret key. Revealing that key brute force is mathematically secure. For example, an AES algorithm that uses a key length 128bit means that there are 2^128 possible available keys. In SCA we target smaller parts of the key to attack so we can take into advantage the information leakage to expose one byte of the secret key at a time (divide and conquer technique). The AES algorithm is a block cipher algorithm. It performs its encryption in blocks of 16 bytes. In PA SCA we target each byte separately. That means that each byte matches one byte of the secret key. Thus, all possible combinations of bits concerning the part of the key that we target each time are reduced to 2^8.

The first step before we perform any kind of attack is to set the target device in such a way that we can measure its power consumption. The main goal is to customize the targeted device in order to measure the whole power consumption with the minimum level of electronic noise. In our case we customized a development board by detaching all grounds except one. On the remaining ground we attached a one-ohm resistor. To successfully measure the total power consumption, we attach the probes of an oscilloscope on both ends of the attached resistor. The customization is shown at figure 13.

The next step aims on finding the part of the power trace which corresponds to the desired part of the targeted algorithm. In our case we target the first S-box transformation of AES algorithm. To achieve this goal, we must use a trigger. In our case we create a signal which we control on the implementation code. When we build our system, we select a specific pin of our target device cpu. The selected pin is controlled through the code. We initialize it with value 0. On the beginning of the desired part of code, in our case the S-box transformation, we set it to value 1. When the S-box transformation ends we set it again on value 0.

The next phase of our attack is to perform a CPA attack against the unprotected software implementation of AES. CPA attack is being explained in detail in section 4. We present the attack methodology of a successful CPA attack in which we manage to acquire the whole secret key byte by byte with the analysis of 1000 power traces. On a typical performance pc the attack we performed lasted on average 4 minutes for every byte.

The last phase of our attack presents the use of a masking scheme on the software implementation of AES. Performing the same CPA attack proves that the masking scheme effectively protects the cryptographic key of AES.

In section two we make a general presentation of all kinds of SCA attacks and we focus on categorization on PA attacks. Section three presents the main categories of countermeasures against PA attacks. The categorization is based on [15]. In the fourth section we demonstrate the first order attack methodology. The fifth section explains the attack through parts of code and a general flow diagram.

2 Side Channel Attacks

Since SCA attacks focus on the physical implementation of an algorithm the main target is the device which integrates the algorithm. Every electronic device leaks information in different forms. Every different aspect of this leakage creates a different adversary. Therefrom we can categorize them in these aspects.

- *Time attacks (TA)*: These kinds of attacks aim on measuring the time of execution on a specific input. This goal can be easily achieved by analyzing the corresponding time on a certain query. TA can reveal important information about the algorithm and the data which are being processed.
- *Electromagnetic attacks (EA)*: Every electronic device when operating emits electromagnetic radiation. Measuring and analyzing this radiation based on electromagnetic spectrum can reveal information about the operation of a device. EA analyzes the power consumption as PA which we are going to refer in detail below.
- *Fault induction attacks (FA)*: FA focuses on stressing an electronic device by erroneous operations. These operations may lead to a security error which may reveal sensitive information. To achieve this kind of attacks the adversary has to be able to induce a fault on a specific time or on a specific location.
- *Optical attacks (OA):* This kind of attack focus on capturing the light (photons) which is emitted when transistors change logical states (0, 1). High resolution cameras are being used to capture these photons.
- *Traffic analysis attacks (TAA)*: This kind of attack analyzes the packet traffic of a network in order to discover its topology. This attack aims on finding and attacking the critical nodes of the targeted network.
- *Acoustic attacks (AA)*: AA focus on recording acoustic emissions of computer peripherals or components (keyboard, CPU, motherboard, printers). Measuring these emissions can reproduce or reveal sensitive information.
- *Thermal imaging attacks (TIA)*: TIA focus on thermal emissions of an electronic device. These attacks usually acquire infrared images to reproduce sensitive information.
- *Power analysis attacks (PA)*: PA attacks aim on measuring the power consumption of an electronic device on a specific time with a specific input. The analysis of these measurements can reveal sensitive information.

2.1 Power Analysis Attacks Categorization

PA attacks are being considered one of the most perilous categories of SCA. PA attacks aim on taking advantage of measuring the power consumption of an electronic device to expose the secret key of an encryption algorithm. These measurements are called power traces and are being used in various ways to reveal sensitive information about the processed data.

We can categorize PA attacks according to the information that we have about the attacked device, the availability of the device which concerns the number of traces that we can acquire and the level of noise that an electronic device has.

1. ***Simple Power Analysis (SPA)***: SPA attack is one of the simplest PA attacks. It requires small number of power traces and it is usually been used when we have lots of details about the device that we are going to attack. This kind of attacks aim on visual inspection of a single or very few traces in order to reveal key-dependent patterns within a trace. SPA attacks can be classified in two main categories according to the number of available power traces, single-shot SCA and multi-shot SCA. In single-shot SCA the adversary uses only one power trace in contrast with multi-shot SCA that uses more traces. The basic idea on both cases is the same. The only difference is that with the second case the bigger amount of power traces of the same input is being used to reduce the electronic noise of the device under attack. In order to understand in depth the SPA

attack we are going to present an example given in [[10]]. In Figure 2 we present a power trace which corresponds to a single run of AES-128 cryptographic operation. After a visual inspection we can clearly conclude that there is a pattern of 10 repetitions. Although we cannot reveal the secret key of AES algorithm we can find out that these repetitions corresponds to the 10 rounds of AES-128 algorithm.

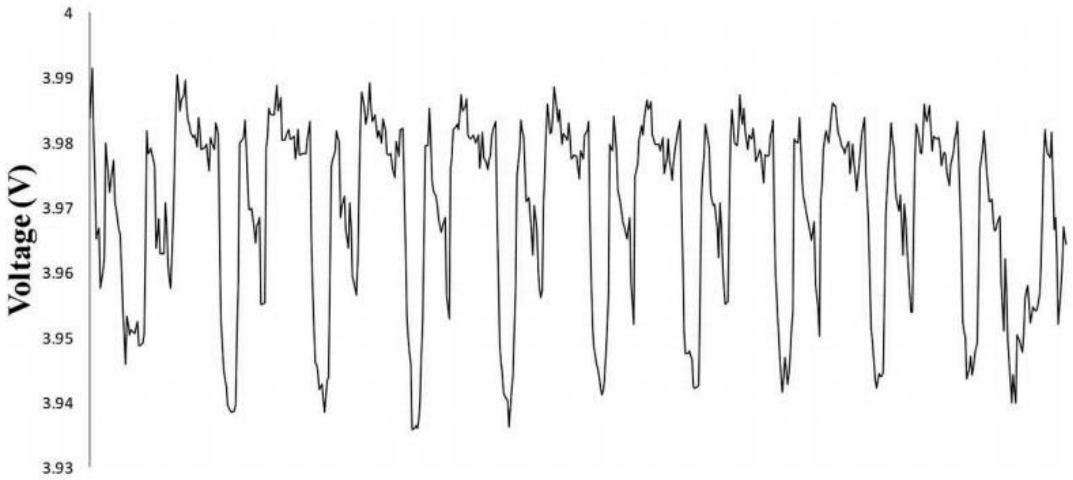


Figure 3 Simple power analysis attack on AES-128

2. **Correlation Power Analysis (CPA)**: CPA attacks are based on the correlation between the power traces and power model hypothetical intermediate values. In this kind of attacks there are two important factors for the successful execution of a CPA attack. The first is the choice of the intermediate value. The second important factor is the power model that the adversary chooses for calculating the hypothetical power consumption. The more precise the power model simulates the actual operation of the target device mechanism the more successful results the attack can have. For example, a hamming distance power model is based on the bit transitions on every state. Hamming distance is ideal attacking on communication bus. Hamming weight model is another power model which is ideal attacking on memory storage. Hamming weight is based on the sum of the existing bits within a targeted 'bit phrase'.CPA attacks can be described into 5 simple steps.

   Step 1: Choosing an Intermediate Result of the Executed Algorithm. This step is quite important because the part of algorithm which computes the intermediate values must include a computation which contains a subset of the secret key.

   Step 2: Measuring the Power Consumption. This step is performed with the help of the digital oscilloscope. Through the system that we describe in subsection Data acquisition system we measure the dynamic power consumption, which we refer to as power trace. In this step we need to store the initial plain text of each encryption which corresponds to each power trace.

   Step 3: Calculating Hypothetical Intermediate Values. In this step we calculate all the hypothetical intermediate values for all the available choices of the part of the key. Since we attack on a smaller part of code the possible available combinations of bits are reduced on a feasible number from computational aspect.

   Step 4: Mapping Intermediate Values to Power Consumption Values. In this step we must choose the power model better matches the power consumption of a targeted device. Through the selected power model, we map data values to power consumption values. There are many different power models that we can choose from depending on the target that we want to simulate.

   - Hamming distance model: power consumption is proportional to number of '1' bits
   - Hamming weight model: power consumption is proportional to transition of bits from 1 to 0 and 0 to 1 (same for both transitions)

- Switching distance model: power consumption is proportional to transition of bits but transition from 1 to 0 and 0 to 1 require different amount of power.
- Signed distance model: power consumption is proportional to charging and discharging capacitance involves a leakage of +1 or -1.

Subsequently we are going to explain in detail Hamming weight power model which we have chosen for out attack.

Step 5: Comparing the Hypothetical Power Consumption Values with the Power Traces. The final step of our attack contains the comparison of the hypothetical power consumption values of each key hypothesis with the recorded traces at every sample point. The indices of the highest correlation values reveal the positions at which the chosen intermediate result has been processed and the key that is used by the device.

3. ***Differential Power Analysis (DPA):*** The target of all PA attacks is to reveal the secret key of cryptographic algorithm. The advantage of this kind of attack is that there is no need of detailed knowledge about the device or the algorithm which is being attacked. In order to accomplish the attack, there is a need of many power traces. DPA attack is almost the same with the CPA attack except the last step (as described above). In the final step a statistical analysis is being performed to be able to compare the hypothetical values with the power traces in order to reveal the secret key of a cryptographic algorithm.
4. ***Higher order DPA (HODPA)*****:** HODPA

   HODPA focus on breaking protected implementations protected by masking countermeasures. In order to manage to break a masking scheme the analysis of multivariate statistics of a signal is needed. An n-th order attack aims on breaking an n-th order masking scheme. In general, an n-th order DPA attack demands n different samples within each power trace that corresponds to an intermediate value. For example, the second order attack is mathematically proven that can break first order masking implementation of AES. The complexity of a DPA attack is based on two factors: The number of the available traces and the length of each trace. The main methodology of the second order attack can be separated into two basic steps based on Oswald and Mangard Herbst and Tillich research [11].

   Step 1**:** Preprocessing the available traces on a fixed interval determined on a well-informed estimation. The first function proposed in [[12]]. In order to conduct an attack, we need two points of a power trace which correspond to the calculation of two intermediate values. In this step we make a targeted guess for the interval of the trace which we assume that both intermediate values are being calculated. Since we are not able to find the exact moment of the calculation, we apply the pre-processing function to all combination of points for every power trace. There are various preprocessing functions which can be selected in order to preprocess the power traces. In [13] the absolute value of the difference of two points was proposed as a preprocessing function. In [14], the usage of the square of the sum of two points was suggested as a preprocessing function.

   Step 2**:** Standard first order DPA attack on the preprocessed power traces. In this step we use a first order DPA attack on the preprocessed power traces.

   In order to understand the HODPA we are going to give a short description of the first step (as be described below) of a second order attack. The prepossessing function that we selected is the absolute differences. As described in [11] the function of absolute differences is based on the mathematical condition:

$$HW(a \oplus b) = |HW(a) - HW(b)|.$$

   Let us assume that a and b are the masked value and the actual mask. The main goal of preprocessing of power traces, is to manage to remove the mask from the masked value. If we include in our power measurement the computation of both mask and masked value the only thing

that we must do to acquire the unmasked values is to tale the absolute difference of all available points within every power trace. The result of this operation will definitely include the hamming weight of initial value before applying a mask.

5. **Template Attacks (TA)**: TA is a special category of SCA because of the precondition that the adversary possesses a device similar to the target device for profiling. The main difference with all other SCA is the fact that the power model which is being used to calculate the hypothetical power consumption is specific for each target device. An advantage of this kind of attack is that since the adversary can program the device at his own will, he is able to target very specific parts of an algorithm. Being more specific on the target area can make the implementation of an attack more feasible and at the same time to prove a vulnerability of the target device.

In order to make a better evaluation of each kind of PA attack we present a table which shows the appropriateness of each attack. To compare all PA attacks, we take into account three cases unprotected implementation of AES and protected with hiding and masking countermeasures.

	Unprotected Implementation of AES	Protected AES with Hiding countermeasures	Protected AES with Masking countermeasures
SPA	✓		
CPA	✓	✓	
DPA		✓	
HODPA			✓

SPA is a really basic attack where bits of the computation can be found by just looking at one power trace. It can be used against unprotected implementations and can have an effect with a single power trace. CPA attack is a powerful attack which can be used against unprotected and hiding protected implementations but in the second case it would require much more power traces in order to remove the (added by hiding) noise. Thus, DPA attacks are suitable against hiding countermeasure. This kind of attacks require large number of power traces and are able to identify differences on power traces. HODPA are made to break masking countermeasure. Since it is mathematically proved that an nth masking scheme can break with an n+1 order DPA attack HODRA are appropriate against masking countermeasure.

3 Countermeasures

The main goal of every countermeasure proposed against SCA is to make the power consumption of the device which implements the cryptographic algorithm independent of the processed data We can classify the countermeasures in two main categories, hiding and masking. To describe in depth these two categories we are going to use the approach of Mangard in[[15]]. In this research there is a classification in three different levels.

- Architecture level which refers to the strategy followed to implement the countermeasures. - Cell level which refers to the structural level of implementations of countermeasures.
- Software – Hardware implementation of a countermeasure.
- Time dimension which focus on the misalignment of the power traces by changing position of each operation within a power trace. - Amplitude dimension which focuses on changing the power consumption characteristics of the performed operations within a power trace.

3.1 Hiding

The hiding countermeasure can accomplish hiding the data dependency of and the power consumption in two ways. The first technique aims on making the power consumption of a device completely random. The second technique focuses making power consumption equal for every operation of a device. In general hiding technique computes intermediate values by creating an artificial difficulty in exploiting information from power traces.

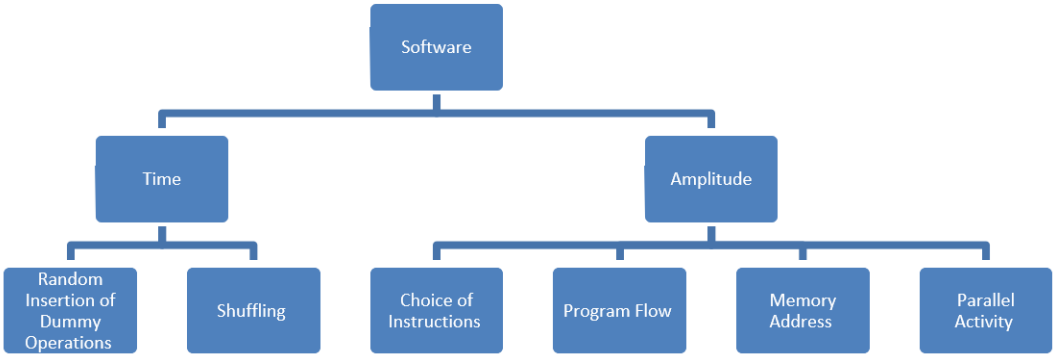


Figure 4 Software approach of Architectural level of hiding countermeasure

Regarding the architecture level of hiding countermeasures, we can classify them in software and hardware level. Concerning software level there are two different aspects. The first aspect is *time*. There are two main categories of this type of countermeasure.

- *Random Insertion of Dummy Operations*: randomly insert dummy operations before, during, and after the execution of the cryptographic algorithm.
- *Shuffling*: randomly change the sequence of those operations of a cryptographic algorithm that can be performed in arbitrary order.

The second aspect of architectural software countermeasure is *amplitude*. We can classify this level in these categories.

- *Choice of Instructions*: Only those instructions should be used that leak small amounts of information.

- *Program Flow*: Changes in the program flow can usually be detected easily when performing a visual inspection of power traces.
- *Memory Address*: Cryptographic devices leak information about specific memory addresses.
- *Parallel Activity*: Increase the noise by performing activities in parallel to the execution of the cryptographic algorithm.

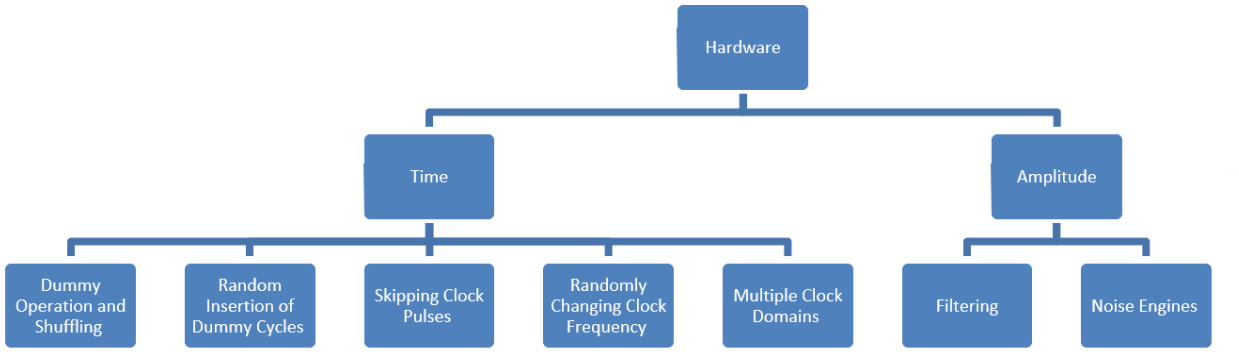


Figure 5 Hardware approach of Architectural level of hiding countermeasure

The hardware countermeasures of the architectural level can also be distinguished on time and amplitude level. Concerning time there can be a classification on the above categories.

- Dummy Operation and Shuffling: There is an implementation in hardware on the same way as it is described in software.
- Random Insertion of Dummy Cycles: While dummy operations can take several clock cycles, the goal of this method is to only insert individual clock cycles
- Skipping Clock Pulses: Insert a kind of filter into the path of the clock signal. This filter randomly skips pulses of the clock signal.
- Randomly Changing Clock Frequency: Generate a clock signal with a randomly changing frequency directly on the cryptographic device.
- Multiple Clock Domains: Several clock signals are generated. Randomly switching between the clock signals.

Regarding amplitude in hardware architectural level we present these two subcategories

- *Filtering*: Remove exploitable components of the power consumption by fitering. A filter is inserted between the power supply pins of the cryptographic device and the circuit that is computing the cryptographic algorithm. Power consumption can be fitered by using switched capacitors, constant current sources, and all other kinds of circuits that regulate the power consumption.
- *Noise Engines*: Generate noise in parallel to the computation of the cryptographic algorithm. Noise engines are typically built based on random number generators.

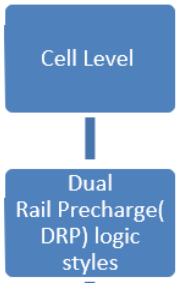
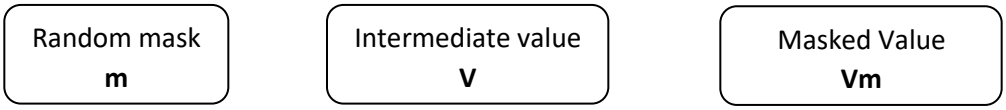


Figure 6 Cell level of hiding countermeasure

The cell level of hiding countermeasure was one of the first counteracts against PA attacks. Cell level countermeasure meand that the logic cells are implemented in a way that power consumption is independent of the prossed data. The technique adopted for this kind of countermeasures is based on Dual-Rail Precharge logic. In DRP logic cells always consume the maximum amount of power in each clock cycle and alter state between precharge and evaluation phase.

3.2 Masking

The masking countermeasure accomplishes to encounter PA attacks by randomizing the intermediate vales of an algorithm. The main advantage of masking technique is the fact that there is no need to change the dependency between the processed data and power consumption. We only need to mask the intermediate values so the actual power consumption which can be measured refers to the masked values. Masking can be either Boolean or arithmetic. A masked intermediate value Vm is an intermediate value v which is concealed by a random value m: Vm = V * m.



Masking technique is secure if Vm is pairwise independent of V and m. The * operation is defined in relation to the operation which is being used by the cryptographic algorithm. Masking countermeasure can provide us sufficient security against PA attacks. If we apply n masks on the same intermediate value, we can make a n-th order masking scheme.

Classification of masking technique regarding architectural level can be assorted in software and hardware aspect.

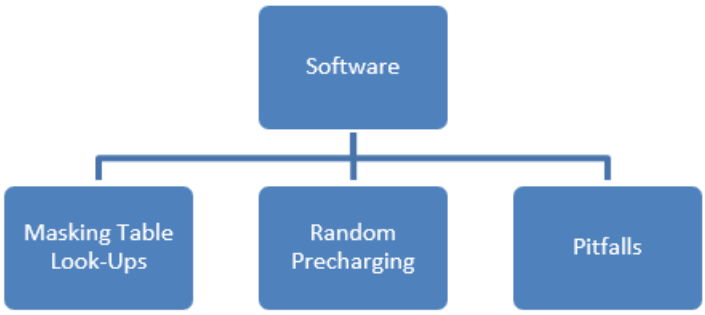


Figure 7 Software approach of Architectural level of masking countermeasure

In software level we can distinguish these classes.

- *Masking Table Look-Ups*: Most modern block ciphers allow implementing the non-linear operations as table look-ups. For each input v of the non-linear operation, the output is stored at the corresponding index in a table T. In masking implementations, the T table needs to be masked. The table is stored in memory where it can be accessed fast. Computational effort and the amount of memory increases with the number of masks that are used to mask the table look-up.
- *Random Precharging*: In this method random values are being loaded or stored before an actual intermediate value is being calculated. Random Precharging, masks the power consumption of intermediate values rather than the intermediate values themselves. It works for operations that leak the Hamming distance power model.
- *Pitfalls*: Power consumption of a device can leak information which can be exploited with the usage of, not only one, but combined intermediate values. Such a combined power consumption of two or more intermediate values can make implementations insecure, even if all intermediate values are provably secure.

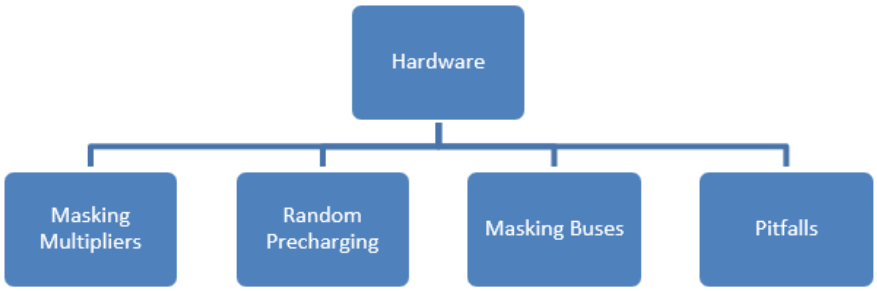


Figure 8 Hardware approach of Architectural level of masking countermeasure

The hardware aspect of architectural level of masking countermeasures can be classified in the subcategories provided below.

- Masking Multipliers: Define a masked multiplier MM (additions are typically easier to mask than multiplications)
- Random Precharging: Random values are sent through the circuit in order to randomly precharge all combinational and sequential cells of the circuit. (requires duplicating the sequential cells)
- Masking Buses: Encrypt the data and address buses that connect the processor of a smart card to memory and cryptographic co-processors.
- Pitfalls: Implementation in hardware level can leak information on the same way as it is described in software.

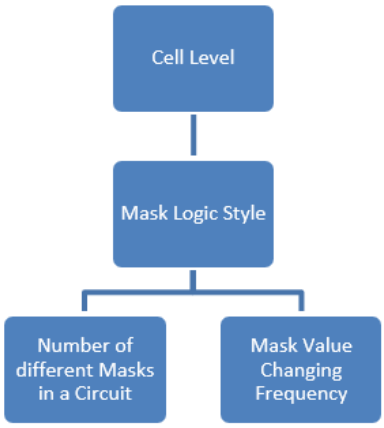


Figure 9 Cell level of masking countermeasure

Regarding cell level of masking countermeasure DPA-resistant logic styles have been proposed. Such DPA-resistant logic styles are usually mentioned as masked logic styles. There are three approaches to build a masked circuit.

1. Distinct mask for each signal.
2. Same mask for a group of signals.
3. Same mask for all signals.

Another aspect of the operating flow of masked logic style is the mask values changing frequency. Masked values are changed in each clock cycle, the rate at which new mask values must be generated is very high. A main advantage of changing the mask values in each clock cycle is that higher-order DPA attacks

4 CPA Attack Methodology

In our attack we focus on revealing the secret key of an AES-128 algorithm. We perform a CPA attack in which we use an oscilloscope to acquire the power traces and MATLAB to send data to the development board which implements AES and conduct the attack. A general description of our attack is being presented on figure 3.

1. Using MATLAB script, a plaintext is being sent from the pc to the development board which implements AES.
2. MATLAB stores the sent plaintext
3. Oscilloscope measures the power consumption
4. MATLAB script saves the measurement of oscilloscope
5. MATLAB script computes the Hypothetical power consumption
6. MATLAB script conducts the CPA attack between recorded power traces and hypothetical values

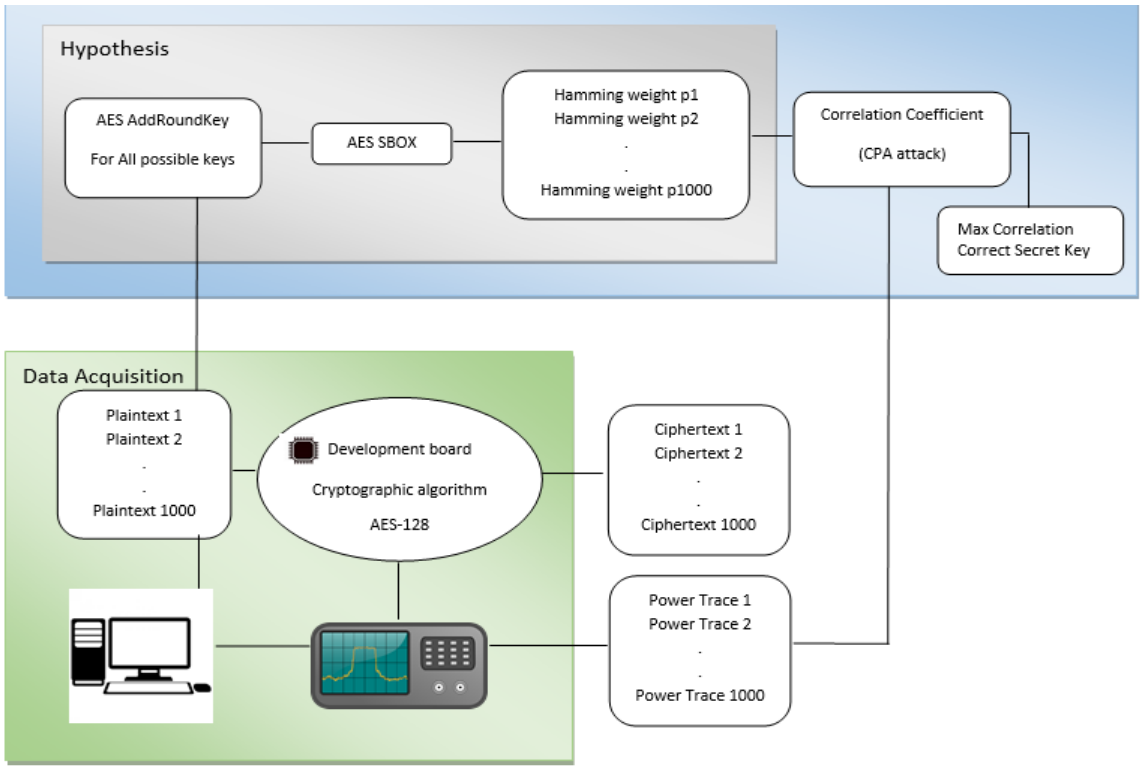


Figure 10 General description of CPA attack.

4.1 Data acquisition system

Our main goal is to acquire data which corresponds on the total power consumption of a device in a precise interval. To accomplish this goal, we need both to control and record the data input of AES. We manage to do this with the help of a MATLAB script. As shown in figure 4 there are two different MATLAB

scripts. The first one sends random data blocks of 16bytes to the AES algorithm through a serial connection and stores the sent data blocks. The second script is used to set the oscilloscope configurations and read the measurements of the targeted device. The power consumption of the device that implements the AES algorithm during encryption of the sent data, is our target. The precise dynamic power measurement of a part of each encryption provides us data that we refer as power traces.

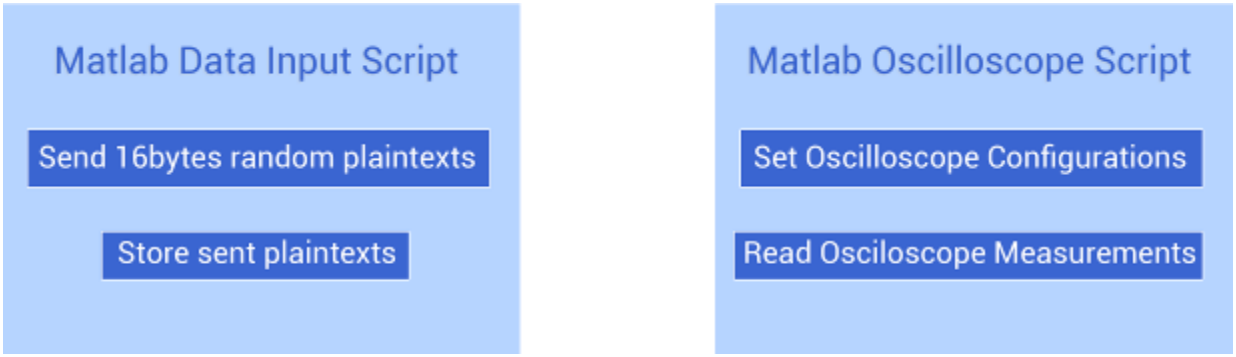


Figure 11 MATLAB Scripts functions

To acquire the power traces, we use a digital oscilloscope. In our system we use an oscilloscope to capture the signal that corresponds to the power consumption of a specific part of AES encryption containing the intermediate value we will use to obtain the secret key. To capture the desirable part of encryption, in our case we target the first round of AES, we use a second signal as a trigger which is being set in the embedded code of the microcontroller under attack which implements AES.

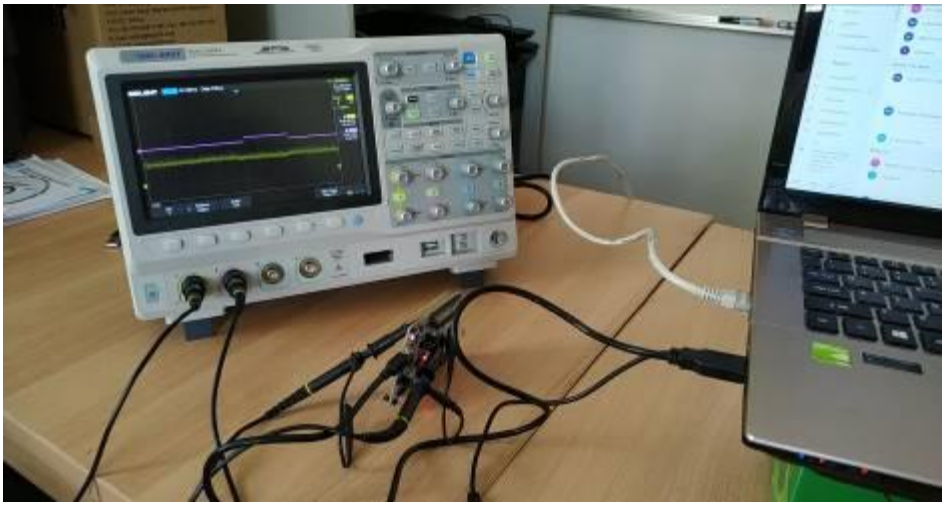


Figure 12 to be done

4.2 Target System

The target system that we use implements AES. The development board integrates an ARM®32-bit Cortex®-M3 microprocessor. In order to measure the dynamic power consumption of the MCU we perform to the low side measurement (GND), by disconnecting all grounds except one. On the remaining ground we attach an 1Ω resistor in series. Thereby if we attach the probes of the oscilloscope on two sides of the resistor, we can measure the total power consumption of the development board. The customization can clearly be seen in figure 4.

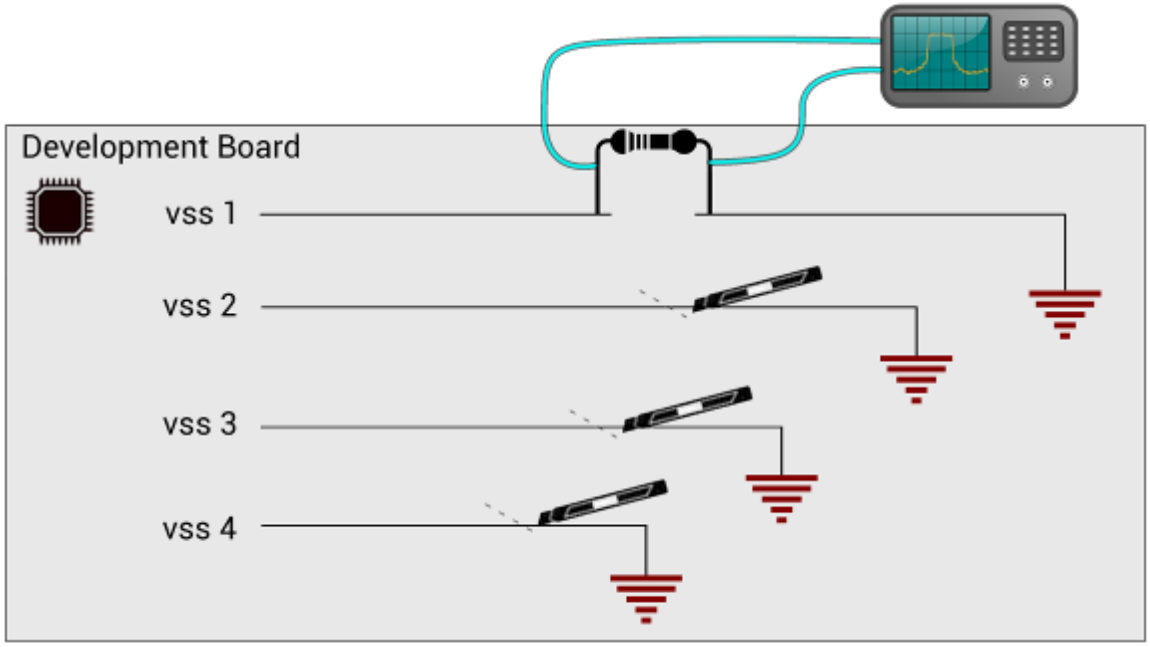


Figure 13 General description of modulation of target device that is being used to implement the AES algorithm

4.3 Hamming Weight (HW) Power Model

The power model that we use in out attempt of describing the actual power consumption of a targeted device is quite important. One of the most important steps of a CPA attack is the attempt to create the power model of the target device. The more precise to the targeted device the power model is the higher correlation between the hypothetical values and the actual produced values could be evaluated on our attack.

The hamming weight of a binary value is the sum of the non-zero bits. The main idea of using HW power model is that the instantaneous power consumption of a device is correlated with the number of bits set to 0 or 1. Every time a bit changes from 0 to 1 or from 1 to 0 a current is required to charge or discharge it. In general, when we measure the power consumption of a device, we are monitoring these bit changes. Since HW describes a power consumption based on bit counts and because of this describes better reads from memory, most of the times makes it ideal for PA attacks.

4.4 Advanced Encryption Standard (AES) algorithm high-level description

AES is a block cipher algorithm. That means that it performs the encryption in blocks. Each block has the size of 128bit and can use three different key lengths. The length of the selected key determines the

rounds that the AES algorithm will perform: 128bit key requires 10 rounds, 192bit key requires 12 rounds, 256bit key requires 14 rounds. In our case we attack on AES-128. We describe the algorithms flow which can be also seen in figure 6.

1. Initial round key addition:
    - AddRoundKey—each byte of the state is combined with a block of the round key using bitwise xor.
2. 9 rounds:
    - SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
    - ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
    - MixColumns—a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.
    - AddRoundKey
3. Final round (making 10 rounds in total):
    - SubBytes
    - ShiftRows
    - AddRoundKey

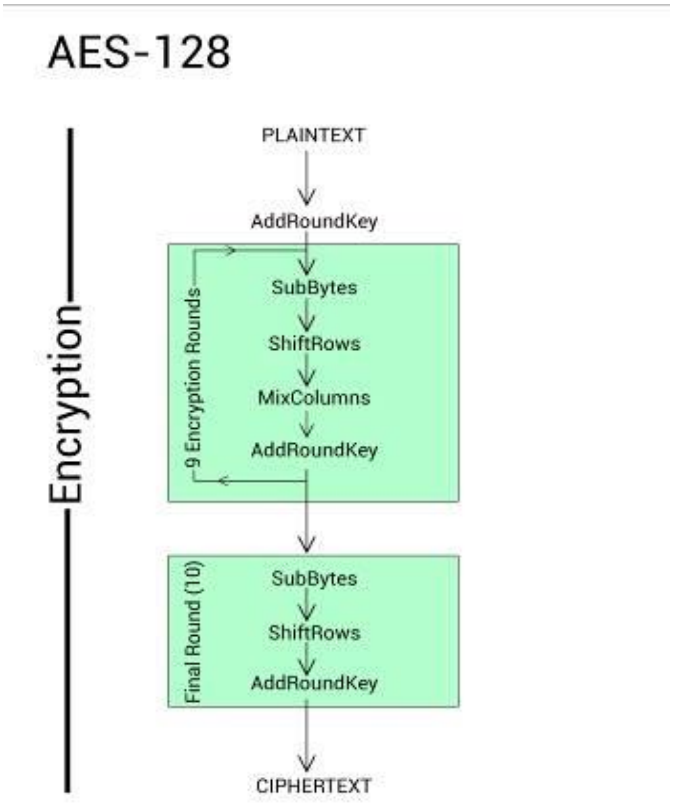


Figure 14 AES encryption flow

4.5 Correlation Power Analysis Steps

As we described in section 2 the CPA attack can be described in 5 steps.

Step 1: Inside this step we must choose the intermediate value that we are going to attack. In our attack we have chosen to attack in the first S-box of AES. The criteria of this choice are based on three main reasons.

- Existence of at least a small part of the secret key inside the computation: The S-box transformation takes us input the first AddRoundKey. AddRoundKey is the bitwise xor of the plaintext and the secret key. Thus, we have inside our computation a part of the secret key.
- Ease of hypothetical calculation of the selected value: Since in the next step of CPA attack we must calculate the hypothetical values of the selected intermediate value, it is more or less helpful to choose an intermediate value which we could be calculated in an easy way. To compute the S-box transformation we just need an xor of plaintext and secret key as input and a matrix to compute the transformation (example is given in next steps).
- Power consumption: The operation of S-box is typically a lookup table which stores values on a given input. We could also choose the first add round key of AES but the power consumption is much bigger in case of S-box which makes the correlation with the acquired traces much easier. As we can see in figure 8 the time for an S-box to complete is much bigger that the AddRoundKey. Thus, the consumed power is bigger in case of S-box transformation.

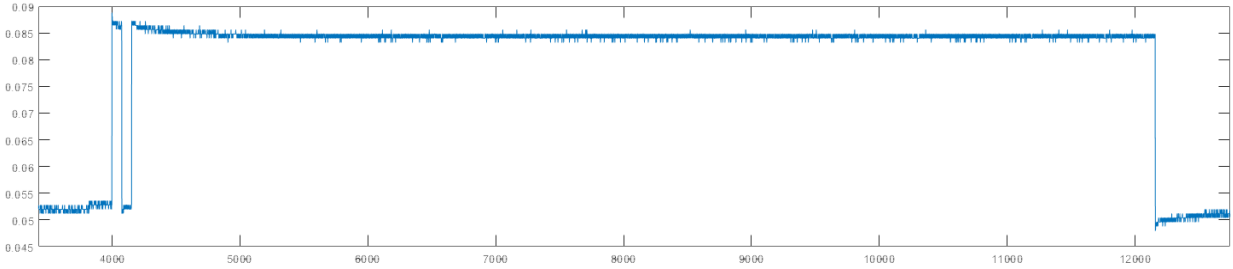


Figure 15 The plot presents the trigger. The first lift of the graph (X axis from 4000-4060) shows the time which AES algorithm needs to complete an AddRoundKey. The second lift of the graph (X axis from 4150-12150) shows the time which AES needs to complete an S-box transformation.

Step 2: In this stage of attack we must measure the power consumption of the target device. To be able to conduct the attack we must store each plaintext in correspondence with the power measurement which creates. As we mentioned earlier, we use a MATLAB script to send via UART port the input to the device. As input of AES we send and store (for usage in next steps of CPA) sixteen bytes of random data. For our successful CPA attack we acquired 1000 power traces. That means that we also stored 1000 plaintexts of sixteen bytes. A short description of our system is being shown on figure 9. First of all, we open a UART connection between the target device and the PC that we use to conduct the attack. Then we set the oscilloscope to be ready to capture a power consumption on the trigger that we have set. In the next step we send the random 16-byte plaintext from the PC to the target device. Then the oscilloscope captures the power consumption of target device encrypting the sent data. From our PC we read and store the captured power consumption that we refer as power trace. To be able to conduct the CPA attack we must store the sent plaintext in correspondence with the power trace that we acquired. This procedure is being followed for each power trace that we need to acquire.

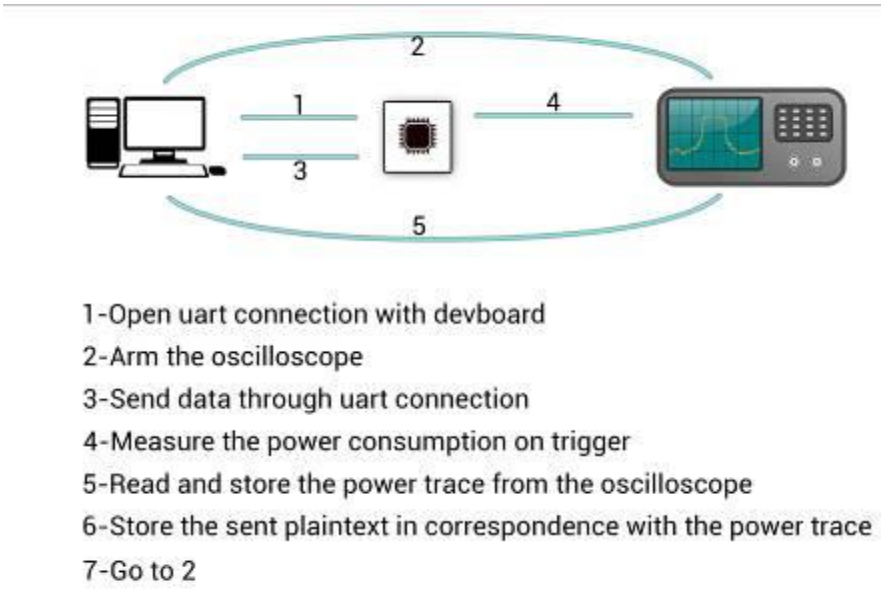


Figure 16 Description of power measurement and data (power traces) acquisition.

Step 3: In this step we must calculate the hypothetical intermediate values. Since we decide to attack the S-box operation of AES we must calculate the hypothetical values of S-box. To understand this part, we will give a detailed example based of table 1.

Random Input Data	Hypothetical key	AddRoundKey (input xor key)	S-box transformation	Hamming weight hypothetical

				value
0xb4	0x98	2c	71 = 0111 0001	4
0x38	0x98	a0	e0 = 1110 0000	3
0xc2	0x98	5a	be = 1011 1110	6
0x5e	0x98	c6	b4 = 1011 0100	4
0xf2	0x98	6a	89 = 1000 1001	3
0xac	0x98	34	18 = 0001 1000	2

Table 1

In the first column we have the first byte of all the random inputs of AES. In the second column we have a possible key (the complete CPA attack must have all possible keys 2^8, for all bytes of all inputs). In the third column we have the hypothetical value of AddRoundKey which is the input xor the hypothetical key. The fourth column calculates the output of the S-box transformation. The transformation can be easily calculated if we take as x of table 3 the first hex digit and y the second hex digit. For example, 2c means that we are going to the S-box table on the third row on the thirteenth column. The returned value is 71. Inside this column we convert the hex value to binary. In the last column we calculate the Hamming weight of the hypothetical S-box output by counting the sum of number of bits set to ones of each value.

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 17 S-box transformation matrix

Step 4: In this step we choose the power model which is more accurate according to our target. As we described in the previous sections the power model that we choose is Hamming weight. The selection of Hamming weight is based on the assumption that the S-box output bits can be linearly correlated with the actual power consumption of the targeted device.

Step 5: The final step is finding the maximum correlation between the power traces and the Hamming weight values of our stored inputs for every byte of the key. To achieve this correlation, we use Pearson's correlation coefficient. This technique can provide us a linear correlation between two inputs. The first input is all the values of every power trace that we acquired in step 2. The second input is the hypothetical Hamming weight values that we calculated in step 3 and 4. The correlation values of Pearson's correlation coefficient can be between -1 and 1. In figure 9 we can clearly see the graphs of two variables with the correlation value above each graph. These examples are several sets of (x,y) points and show that there is only the linear correlation which is being calculated and not the kind of correlation.

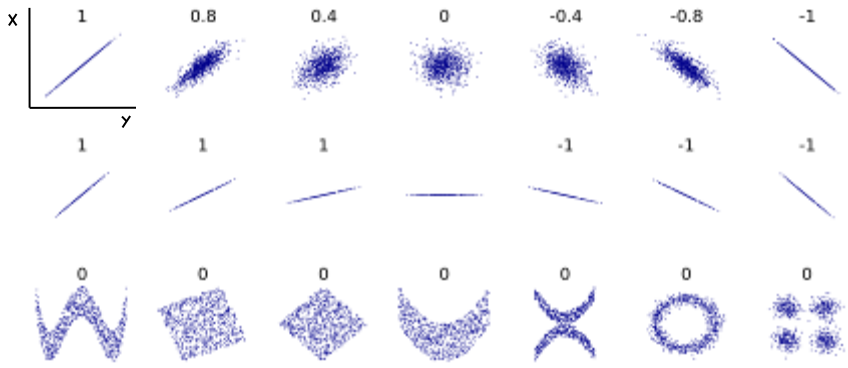


Figure 18 Pearson's Correlation Coefficient examples (sets of x,y variables)

In our attack the correlation is between matrix A and B. Matrix A in x axis has different traces (in our case 1000) and in y axis has the time in points (for our attack every trace has 14000 points). Matrix B in x axis has different plaintexts (which correspond to the traces of matrix A) and in axis y has the hypothetical power consumption that we calculated for all possible keys, that means that we have 256 columns. Each column corresponds to different key value. The result of this correlation can provide us the column of matrix B with the max correlation value. This value reveals us the column which corresponds to the key which has been used for the hypothetical calculations. We repeat this operation for each of the 16 bytes of our input and we finally manage to acquire the complete secret key of AES.

Number of traces	Time points
	Matrix A

Number of plaintexts	Hypothetical power consumption for all possible keys (2^8)
	Matrix B

In Figure 11 there is the result of a successful CPA attack. The figure shows the attack on the first byte of an AES-128 implementation. The spike refers to max correlation of trace and hypothetical value based on the correct key guess.

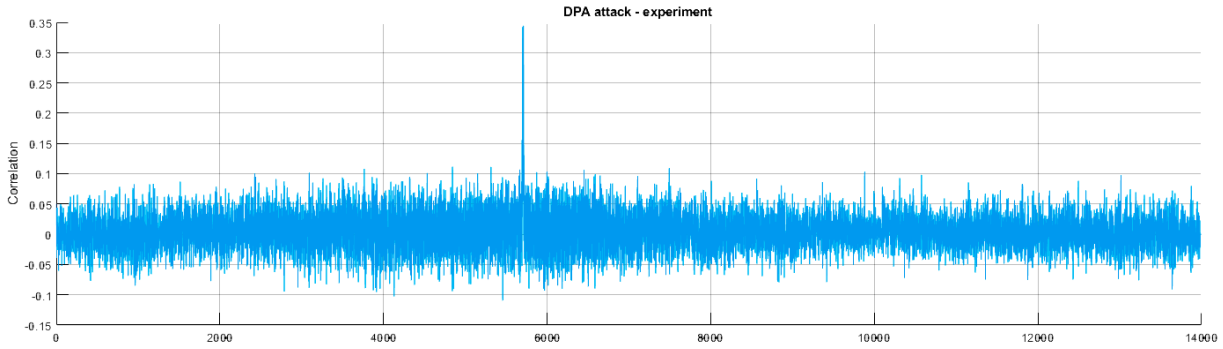


Figure 19 Example of successful CPA attack. X axis presents the time points of the traces which the max correlation was found. Y axis presents the correlation of power consumption and hypothetical values.

5 Code and flow diagram

5.1 Code

In this section we are going to present a pseudocode of the attack.

In our attack code we have three matrices:

1. Matrix with the 16 bytes plane texts corresponding to the recorded power traces, which has different plane text on each row and different byte on each column. This matrix refers to *data* on the pseudocode.
2. Matrix with the traces which has different trace on each row and time points on each column. Our trace matrix has 1000 traces and 14000 sample points. That matrix refers to *traces* on the pseudocode.
3. S-box transformation matrix of AES. That matrix refers to *V* on the pseudocode.

The first loop on our attack in line 5 of figure 12 has to do with the byte that we are going to attack. Since we attack on AES-128 the block size is 16bytes.

In the next step of our attack we calculate the hypothetical power consumption. The AddRoundKey is calculated by making xor between all possible keys (*K*) with the byte of plain Text we attack. Then we compute the Sbox transformation with the operation of SubBytes.

The last step of out attack is the part that we calculate the correlation of the recorder power traces and the hypothetical power consumption of every possible key. In line 10 we compute the correlation by the operation *correlation*. The variable *mp* represents the time points which each power trace is being measured. The *RowOfR* operation finds the row which the max correlation was found (refers to the correct key).

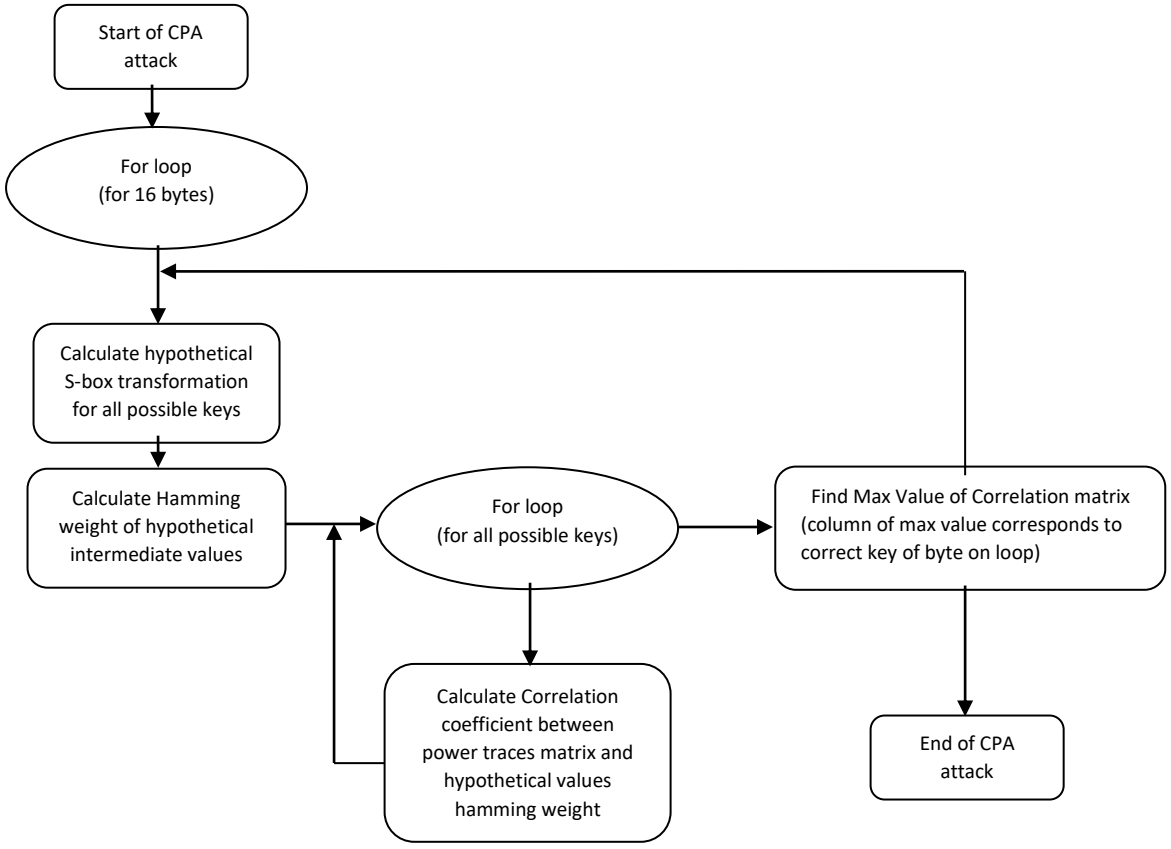
After the calculation of the correlation coefficient of the previous step we find the maximum correlation which corresponds to the correct key on the specific byte that we chose to attack.

```
1  start CPA attack
2  traces= load(PT)
3  data = load(PlnT)
4  K=0 to 255
5  for a=1 to 16
6      V=SubBytes(K xor data(a))
7      HW=HammingWeight(V)
8      for 1 to 256
9        for 1 to mp
10         R=correlation(HW, traces(a))
11     key=RowOfR(max(R))
12 end CPA attack
```

Figure 20 pseudocode CPA attack

5.2 Flow Diagram

In this section we present a high-level flow graph of the CPA attack that we performed.



The first loop is repeated 16 times as many as the number of the bytes that we attack. As we mentioned in previous section AES is a block cipher algorithm with block size 16 bytes. Inside the loop we need to calculate all the hypothetical S-box transformations for all possible keys. Since we attack on a single byte the length of the key part that involves in this transformation is also one byte. That means that we follow the steps described in section *CPA steps* in third step. That operation also includes the calculation of hamming weight of all intermediate values. The next step is to loop over all possible keys (2^8=256). Inside every loop we calculate the correlation coefficient between the power traces matrix that we acquire with the help of the oscilloscope and the hypothetical values hamming weight that we evaluated in the previous step. In the final step we find the max value of correlation matrix. The column which the max value is found corresponds to the correct key of byte that we attack (determined by the first loop). After the repetition for all 16 bytes we manage to acquire the whole 128bits of the secret key.

6 CPA Attack Against Protected Implementation

In order to protect AES-128 implementation against the CPA attack that we conducted we tried to perform the same attack against a protected implementation. The countermeasure which we applied on the AES encryption is masking technique. The masking countermeasure was implemented in software level. Thus, on the first part of AES encryption random masks were applied on the input data and the secret key. The nonlinear part of the encryption is based on GitHub project []. We performed the attack in two different subsets of power traces. The number of acquired power traces were 1000 and 5000 in the first and second subset respectively. As we can see in figure 21 and 22 in none of the cases can be found a high correlation as shown in figure 19. We are not able to find any spikes referring to a hypothetical value containing a part of the secret key.

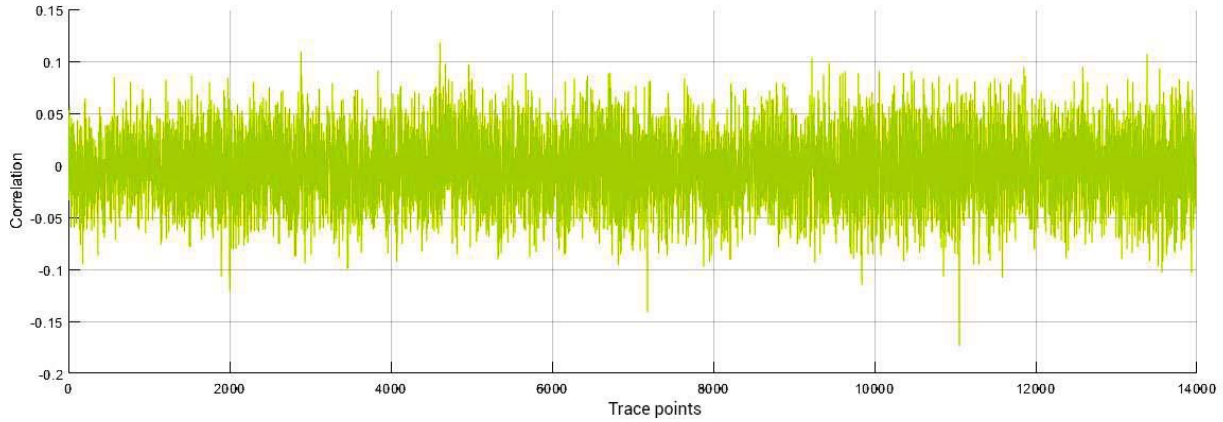


Figure 21 1000 traces sample CPA attack on first byte of AES-128

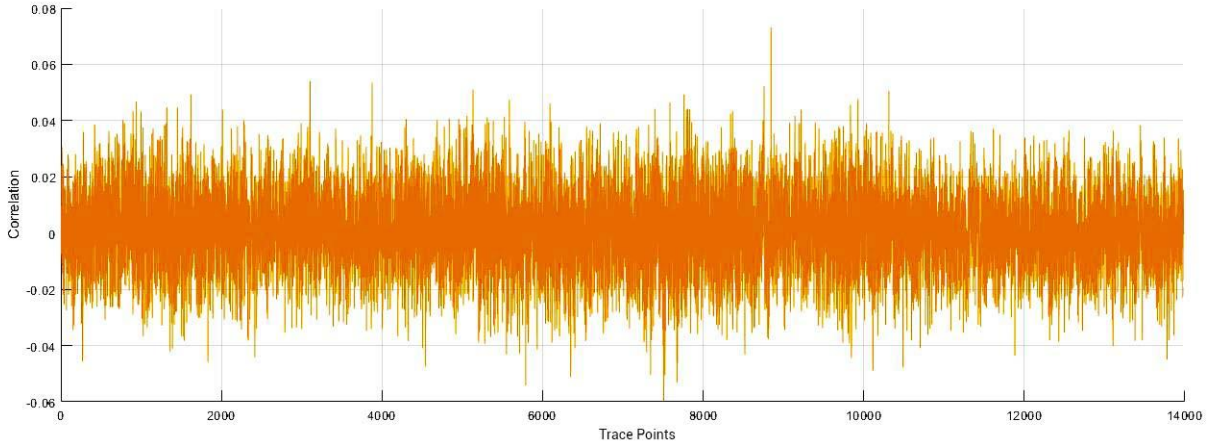


Figure 22 5000 traces sample CPA attack on first byte of AES-128

7 Conclusion

On the beginning of this master thesis we made a general description of SCA attacks. We presented a general categorization concerning the type of information leakage which an adversary could take advantage of. These were attacks such as timing, electromagnetic emission, optical, fault induction, acoustic, thermal, power analysis attacks. Then we focused on PA attacks and made a categorization on all different kind of PA attacks. At that time, we presented countermeasures against PA attacks so we can have complete view of PA concept. We analyzed the two main categories of PA countermeasures, hiding and masking. We presented these techniques in architectural and in cell level.

In order to be able to understand the PA that we conducted we made a short description on different parts of our attack both in software and hardware. Regarding hardware we described the data acquisition system that we used to record the power traces. We also made a short description on the target system which implements the AES algorithm. In software level we made a high-level description of AES algorithm and we presented the power model that we used to describe the power consumption of the target device.

At the next step we presented the methodology that we used to conduct a CPA attack. We presented a description based on [15] which divides the attack in five steps. After we describe in detail these steps, we show a figure with the results of a successful CPA attack.

In the last section we created a pseudocode which describes the attack that we conducted. In order to understand in a more general aspect the attack we created a flow diagram of the code that we used to make the CPA attack. Furthermore, we perform the same attack that we conducted against a masked protected implementation. We ascertain that the masking countermeasure provides security against a CPA attack.

AES implementations against SCA are and will be an active research field. Many attempts have been made for securing the AES implementation against known attacks but more or less the resistance against SCA is in fact matter of available power traces, time and resources. The quantity and the quality of these factors combined with emerging technologies such as machine learning, put a threshold on what we can present as secure.

1. Dusart, P., G. Letourneux, and O. Vivolo. *Differential Fault Analysis on A.E.S*. 2003. Berlin, Heidelberg: Springer Berlin Heidelberg.
2. Bar-El, H., et al., *The Sorcerer's Apprentice Guide to Fault Attacks.* Proceedings of the IEEE, 2006. **94**(2): p. 370-382.
3. Tajik, S., et al. *Laser Fault Attack on Physically Unclonable Functions*. in *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. 2015.
4. Kelsey, J., et al. *Side channel cryptanalysis of product ciphers*. 1998. Berlin, Heidelberg: Springer Berlin Heidelberg.
5. Spreitzer, R., et al., *Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices.* IEEE Communications Surveys & Tutorials, 2018. **20**(1): p. 465-488.
6. Kocher, P., J. Jaffe, and B. Jun. *Differential Power Analysis*. 1999. Berlin, Heidelberg: Springer Berlin Heidelberg.
7. Mangard, S. *A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion*. 2003. Berlin, Heidelberg: Springer Berlin Heidelberg.
8. Brier, E., C. Clavier, and F. Olivier. *Correlation Power Analysis with a Leakage Model*. 2004. Berlin, Heidelberg: Springer Berlin Heidelberg.
9. Blömer, J., J. Guajardo, and V. Krummel. *Provably Secure Masking of AES*. 2005. Berlin, Heidelberg: Springer Berlin Heidelberg.
10. Lo, O., W. Buchanan, and D. Carson, *Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA)*. 2016. 1-20.
11. Oswald, E., et al. *Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers*. 2006. Berlin, Heidelberg: Springer Berlin Heidelberg.
12. Chari, S., et al. *Towards Sound Approaches to Counteract Power-Analysis Attacks*. 1999. Berlin, Heidelberg: Springer Berlin Heidelberg.
13. Messerges, T.S. *Using Second-Order Power Analysis to Attack DPA Resistant Software*. 2000. Berlin, Heidelberg: Springer Berlin Heidelberg.

14. Waddle, J. and D. Wagner. *Towards Efficient Second-Order Power Analysis*. 2004. Berlin, Heidelberg: Springer Berlin Heidelberg.
15. Mangard, S., E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. 2007: Springer-Verlag.