Tampere University

Placido Bottari

ISOLATING ELECTROMAGNETIC LEAKAGE FOR SIDE CHANNEL ANALYSIS

Master's Thesis, Computer Sciences
Faculty of Information Technology and Communication Sciences

Examiners:
Professor Billy Bob Brumley
Senior Research Fellow Alejandro Cabrera Aldaya,
Doctoral Researcher Iaroslav Gridin
June 2023

ABSTRACT

Placido Bottari: Isolating Electromagnetic Leakage for Side Channel Analysis
Master's thesis, Computer Sciences
Tampere University
Master's Programme in Information Technology: Information Security
June 2023

Electromagnetic (EM) side-channel analysis (SCA) is a procedure used to find weaknesses in the implementation of cryptographic devices allowing a user to extract secret data. An EM scan of a device is performed in specific areas of a cryptographic device, more precisely where the cryptographic operations are executed. EM SCA techniques are powerful procedures in terms of effectiveness and non-invasive functions for the Internet of Things devices, these attacks do not require any physical modification of the target system.
This work describes the assembly of an automated measurement EM SCA platform, assembled with components that are possible to purchase on the web. The measurement bench allows to perform precise EM SCA analysis to define points of interest using the signal-to-noise ratio (SNR) or test vector leakage assessment (TVLA) to define the acquired EM SCA traces without the need of an expert in signal processing and radiofrequency. This work also presents a measuring algorithm developed in MATLAB to automatically locate high leakage points defining an area of attack. When the area containing the hot spots that release a high level of EM emissions is defined, a further analysis is performed to collect the EM traces and study the Hamming weight applying Differential EM Analysis (DEMA), Correlation EM Analysis (CEMA), and Differential Frequency-based Analysis (DFA) by the knowledge of the Hamming weight collected.
The contribution of this work explores the tampering needed to capture EM traces from a cryptographic algorithm. The proposed system allows automation of the process of localizing, recording, and analyzing the EM leakages emitted by the execution of a cryptographic algorithm through a heatmap that defines a region of attack.

Keywords: cryptography, side-channel analysis, measuring device, Hamming weight, electromagnetic traces

PREFACE

This work was supported by the Network and Information Security Group (NISEC) as part of the ARROWSMITH project: the Cybersecurity Research Award granted by the Technology Innovation Institute (TII) in UAE and Technology Innovation Institute's Secure Systems Research Center (SSRC) in UAE. The present thesis has been carried out during the spring semester of 2021 at Tampere University Finland and presented at the Department of Computer Science at Tampere University Finland.
This work was conducted by the author under the guidance and collaboration of Professor Billy Bob Brumley, Senior Research Fellow Alejandro Cabrera Aldaya, Doctoral Researchers Iaroslav Gridin, and Sohaib ul Hassan, at Tampere University of Technology.
I would like to extend my gratitude to Professor Brumley for his guidance and collaboration, Senior Research Fellow Alejandro Cabrera Aldaya, Doctoral Researchers Sohaib ul Hassan, and Iaroslav Gridin for supervising and providing feedback on this thesis.
For their support, my family, friends, colleagues, peers, and study coordinator.
A special thanks goes to Francesco Comito, Carolina Ceruti, and Angelica Marini for being always present.

In Tampere, June 28, 2023

Placido Bottari

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AES	Advanced Encryption Standard
$Area_a$	Area of acquisition
σ_W^K	Correlation of the Hamming weight of the predicted value at the key guess
$\rho^K(t)$	Correlation factor
$\sigma_C(t)$	Correlation of the sample C
CEMA	Correlational electromagnetic analysis
DUA	Device under attack
DUT	Device under test
DEMA	Differential electromagnetic analysis
DFA	Differential frequency analysis
μ_W^K	Differential mean value of the Hamming weight of the key guess
$\mu_C(t)$	Differential mean value of the correlational analysis
$D^{j,K}(t)$	Differential trace
$P_i^b(f)$	Differential traces under the Fourier domain
$E_{C,W}^K$	Expected value of the correlational analysis Ci(t) as correlational factor at time t based on the key guess.
EM	Electromagnetic
FFT	Fast Fourier Transform
W_i^K	Hamming weight of the predicted value
H_K	Hamming weight measured also as heigh of the differential.
IoT	Internet of Things
K	Key guess
k	key byte
MHz	Mega Hertz
mm	Millimeters
MTD	Measurements to Disclosure
PCD	Personal Computer Device
p_i	Plaintext
POI	Point of interest.
RoA	Region of attack
$C_i(t)$	Sample at time t i^{th}

SCA	Side-channel analysis
SNR	Signal to noise ratio
$D_S^{j,K}$	Significant Differential trace
SEMA	Simple electromagnetic analysis
TVLA	Test vector leakage assessment

1. INTRODUCTION

Unintentional transmission of data is unavoidable in the majority of the devices of daily use; electronic hardware, in fact, emits electromagnetic (EM) fields and power signals, creating new communication channels whose users are often unaware of. Those channels are defined as side channels, as they deviate from the conventional way of transmitting information between the two end-users and represent a main vulnerability of the device. Some examples of side channels are power consumption, computation timing, or EM emissions; those leakages may be crucial during the execution of cryptographic algorithms. A cryptographic device during the execution of encrypting/decrypting operations, renowned to be mathematically secure, can be effectively exploited if it is not implemented correctly [1].

In particular, EM side-channel emissions can be captured by a malicious user in terms of EM traces or power signals from a device during the execution of a cryptographic process. The cryptographic algorithm executed by a processor will create a current switching on the chip, this switching current will possess key-dependent information. Therefore, power side-channel analysis (SCA) can reveal information regarding the key. Furthermore, the current drawn and switched by the operating chip can also release complex EM emissions through the interaction with active chip components. These emissions are generated by printed circuit board, package, and any other components on the device. It can be useful to remark that, the conducted analyses have shown that the information about the key operation that leaks out through EM interactions is sufficient to compromise the security of the key itself.

The electromagnetic samples collected by the emitting device over a certain period of time and at a certain rate are called traces. The collection of traces captured in one run from the scope is called capture or acquisition, where the scope defines the ranges of orientation of the measurement.

One of the main exploitations of EM side channel vulnerability is the electromagnetic side-channel analysis (EM SCA), which is a technique that allows finding implementation weaknesses in cryptographic devices to extract secret data. In previous years, the EM scanning of devices took place in specific areas that perform cryptographic operations by analyzing the minimum traces Measurements to Disclosure (MTD). The captured EM traces require fulfilling an average number of acquired measurements including the correct target result in order to reach a success rate larger than 50% and been computed as MTD. By means of MTD operations a user can ensure that the probability of collecting

the target result is achieved with a success of more than 50% of the attempts. A single MTD value could not be fully independent, while it is computed on a disjoint measure subset of the whole measure set (because MTD are not originated by statistically independent experiments). The acquisition of a measure is made of multiple operations that are performed until reaching the target result in the target operation [2].
For this reason, special studies on EM SCA are conducted, because these procedures provide a powerful outcome in terms of effectiveness and non-invasive techniques for Internet of Things (IoT) devices [3].
In cryptography, EM SCA attacks are identified as a threat to the security of information and communication systems as they make it difficult to keep the principle of confidentiality and integrity of sensitive data intact. Power scanning can be caused by an attacker to acquire power signals or electromagnetic emissions while performing cryptographic operations on a device. Power scanning is used to perform power analysis attacks [4].
The basic principle of power analysis attack is to reveal a cryptographic device key through the analysis of its power consumption [5]. The power consumption under investigation for this operation is data-dependency and operation-dependency. Power analysis attacks are conceived on the exploit of instantaneous power consumptions in a cryptographic hardware depending on the processed data and performed operations [6].
Differently from power attacks that are based on the study of power consumption, EM SCA attacks refers to the analysis of EM traces captured on a certain area of a cryptographic chip [7]. EM SCA requires that the attackers choose the exact location where to perform the attack in the device under attack (DUA) to record the EM traces using an EM probe [8]. This choice can have a significant impact in terms of the effectiveness and efficiency of the attack if the probe is positioned in correspondence of the access memory of the S-box. These EM SCA attacks are effective, because they do not require any physical modification of the system to be targeted for an attack. Moreover, the benefits are that the attacker can choose the location where the maximum amount of SNR of information leaks out. This attack, if compared with power SCA, requires an additional number of challenges to be overcome.

A powerful, commonly used EM SCA technique to analyze and extract information from a one trace set is the Simple Electromagnetic Analysis (SEMA). SEMA allows to collect information on the data under investigation in terms of amplitude of the Hamming weights, which represent the number of bits that have been switched from 0 to 1 visible in a scope measured as high amplitude Hamming weights. But SEMA presents some limitations in the noise, and to for this reason it is necessary to proceed with more advanced analysis defined as Differential EM analysis (DEMA), Correlation EM analysis

(CEMA), and Differential Frequency Analysis (DFA), whose measurements are taken while a cryptographic algorithm is running on the device under test (DUT). Thus, each measurement is known as an EM trace and these are correlated to a leakage model, such as Hamming weight or Hamming distance of data at a particular point in an algorithm, assuming a subset of the secret key. In a successful EM SCA attack, the hypothesis that results in the maximum correlation corresponds to the secret key [9]. By attacking the hidden key incrementally, for example, one byte at a time for Advanced Encryption Standard (AES), the entire secret key can be recovered in less time than brute-force attack or other cryptanalysis methods.

The EM SCA attacks are conceived with the knowledge that key-dependent operations create key-dependent EM traces, which contribute to the side channel signal emitted by the cryptographic device, with the negative effect that the single operation is not conducted singularly but in combination with other operations that induce algorithmic noise [10]. EM SCA attacks have been used successfully on PCs and smart cards, but in order to be effective many challenges have to be overcome [11].

1.1. Challenges

The EM SCA techniques challenges are here presented:

1. EM signals, before being processed, need to be transformed using EM analysis. This analysis is necessary to reduce the noise floor, but this condition affects the measurement reducing the resolution of the measurements, more advanced devices can be effective but increase the costs of the measurement devices. This work implements differential EM analysis in the time domain, this implementation makes it possible to control the level of noise floor without damaging the precision during the measurements. Furthermore, this work makes it possible to perform the frequency based differential electromagnetic analysis through the frequency domain to improve the performance [9].
2. Differently from power attacks, EM attacks require that the attackers choose the exact location where to perform the attack in the device under attack (DUA) to record the EM traces. This choice may have a significant impact in terms of the effectiveness and efficiency of the attack. This work describes an implementation of an automated algorithm that can help the user to define the exact location where to perform the EM SCA analysis.
3. EM SCA is a technique that can present high costs in terms of implementation, measurement performance, and data storage. The cost of the implementation

can be low or high, depending on how much the user wants to go further into it. The main costs are probes, oscilloscopes, and/or EM test bench.

4. EM SCA measurements on the device are directly proportional to the increase in the size of the cryptographic chip to be analyzed. Therefore, the different analysis techniques proposed require that the user is an expert in signal processing and electromagnetic theory, in order to discover the location of where to orientate the scope to acquire the EM traces.

1.2. Goals of the project

Under the presented challenges, the goal of the thesis is to assemble a sniffing bench, made with the following characteristics:

- **Noise reduction:** To capture the EM emissions, it is necessary to reduce the noise floor, for this reason it is necessary to reduce the internal defenses of the cryptographic device and simulate tampering of the devices. Through the emulated tampering, it will be possible to reduce all the undesired operations performed by the device while running the cryptographic algorithm. Moreover, the tampering procedure will allow to insert a trigger into the algorithm and discover the information for `round 1` of AES encryption to record EM SCA leakages using an iterative number of looped acquisitions. This operation gives the output to the oscilloscope when it is time to acquire the measurements. This implementation makes it possible to control the level of noise floor without affecting the precision during the measurements.

  The core of this work is to assemble a measuring platform moved by three stepper motors, where each engine moves a coordinate direction length (X coordinate), height (Y coordinate), along the surface and width (Z coordinate) to place an H-probe at the exact location to capture the EM emissions.
- **Measurements:** The sniffing platform can perform precise EM side-channel scans, to find point of interests (POIs) using Signal to Noise Ratio (SNR) or Test Vector Leakage Assessment (TVLA) of the EM SCA traces. The Sniffing Platform in this case will be a scanning device with a near-field H probe to detect the EM SCA leakage of the device under test (DUT). With the increase of the Internet of Things (IoT), the security of many devices becomes critical, as they make it difficult to keep intact the principle of confidentiality and integrity of sensitive data. The developed device can be used to test IoT device implementation vulnerability to EM SCA attacks.

- **Cost-effective:** The system is designed with components that are possible to find available from various sellers on the web, automated and efficient, in order to measure the EM SCA leakages, even for users that are not experts in this field. Furthermore, the time needed to acquire all the necessary EM traces was monitored and the process to forward them, maintain at average level the memory usage to store the data.
- **Requirements:** The device also requires a measuring system developed in MATLAB to automatically perform exhaustive searches on the whole cryptographic chip or choose an arbitrary location where it is possible to collect EM traces. This process allows the user to detect points of high loss defined as Region of Attack *(RoA).* In that region will be performed EM SCA analysis to study the Hamming weight and apply DEMA, CEMA, and *frequency-based DEMA*. This procedure will help the user providing an automated algorithm in the process of locating where to place the scope to acquire the EM traces. Also, the user can analyze with the help of different operational script the captured EM emissions, mitigating the problem that the user is not in possess of a deep knowledge of signal processing and electromagnetic theory [12].

1.3. Contribution

In recent years, SCA measurements are beginning to be developed as forensic analysis to retrieve information of forensic value. This analysis is challenging because desktop and laptop computers and smartphones, defined as Personal Computer Device (*PCD)*, present limited interfaces to retrieve significant information. Nevertheless, even if this kind of attack is difficult to perform, it results as a threat to IoT devices. For this reason, this project contributes to the process of reducing the amount of tampering for end-to-end devices [13].

In the second chapter, EM SCA will be explored, presenting in more detail how it differs from normal SCA, how it is executed, how it is measured, the necessary tools to characterize the measurement, and the theory involved [14].

The third chapter explores materials and methods and presents an implementation of EM SCA. The theory discussed in the state of arts is applied [12]. The chapter shows how to set up the measuring bench and the challenges necessary to achieve the assembling of the working device [5].

The fourth chapter presents the obtained results starting with the heatmap to understand how to discover the RoA, then how to evaluate the Hamming weight necessary to discern

from one operation to another. Hamming weight allows to understand if the operations with the key has been involved or not if evaluated with DEMA, CEMA and *DFA*. The latter is studied as a lower case of misalignment of traces.
Finally, a detailed conclusion of all the obtained knowledge is drawn.

2. OVERVIEW ON DIFFERENTIAL SCA

This chapter presents the application methods for differential side-channel analysis and starts with a description of the measurement instruments used to acquire the traces, followed by necessary frequency analysis calculations needed to process the information obtained and a brief explanation of how to set up an attack on a target device. The work discussed throughout the chapter focuses on implementing a simple and inexpensive way to create an electromagnetic measurement test bench. The acquiring device can identify a point on the targeted hardware where the leakage amount is high enough to automatically detect side channel traces, perform the necessary differential and correlation operations, and finally evaluate the EM results. Furthermore, this work enables even non-expert users of embedded systems to perform these analyses.

The EM SCA have been derived by the first idea of Differential Power Analysis (DPA), that was presented by *P. Kocher in CRYPTO99* [15], that states as follows: "*DPA can be used to break implementation of almost any symmetric or asymmetric algorithm. It has been used to reverse-engineering unknown algorithms and protocols by using DPA data to test hypotheses about a device's computation process. (It may even be possible to automate this reverse-engineering process)*" [9] [1]. After a few decades, Differential EM SCA has been developed as a technique that allows the investigation of unintentional information leakage, in the different form of *"emissions"* [16] as shown in Figure 1: here, different kinds of involuntary losses that could lead to vulnerabilities are exposed. It is possible to state that: " *although all electronic devices emanate electromagnetic waves, it was not known whether information in the waves emanating from a highly complex device could be used to determine the secret key of a cryptographic computation "* [16]. *Tromer (2004)* [1]. The general rules of cryptography say that the secret key must always be protected in any embedded system, to avoid consequences such as identity theft, financial loss, and many more. In the next section, the Differential EM SCA will be discussed, showing how a device can be exposed to Differential EM SCA attacks, and how it is possible to acquire information with a probe from a device.

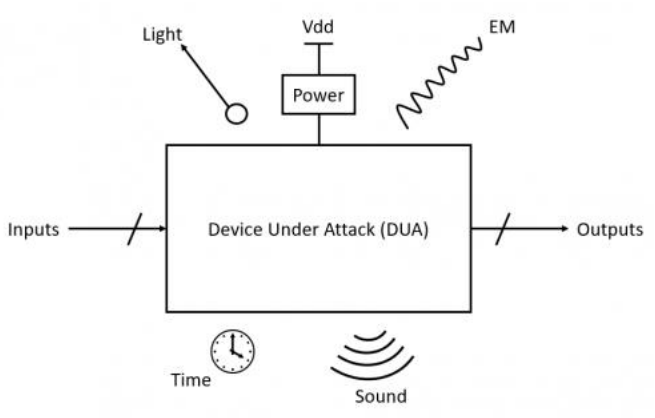


Figure 1. Types of unintentional losses.

2.1 Differential EM SCA

To introduce the Differential EM side-channel theory it could be useful to investigate in more detail the general discussion of the behaviour of the circuit responsible for creating EM wave dispersion.

The cryptographic algorithm executed by a cryptographic processor will create a current switching on one of the die inside the device, where this switching current will possess key dependent information because the current drawn and switched by transistors generates complex EM emissions through the interaction with capacitance and inductance. Therefore, those emissions can reveal information regarding the key. These interactions are not limited only to capacitance and inductances on the die, but even from printed circuit boards, packages, and other components in the device. However, even if the data obtained through the EM leak is difficult to analyze, the information that can be obtained is high enough even if the leak is low, then it is possible to perform an analysis on the key processing. It can be useful to remark that the conducted analyses have shown that the information that leaks out through EM losses about processing with the key is sufficient to compromise the security of the key.

The first analyses that have been used to extract key information from the chip are the Simple EM Analysis (SEMA) and Differential EM Analysis (DEMA).

SEMA is the method used to analyze one trace from the DUA and to define the amplitude of the Hamming weights of the acquired measurements. DEMA is generally used to derive and provide knowledge of the input data (plaintext) and links how the plaintext and key are involved in exclusive-OR-ed computation already characterized into index table. Furthermore, these techniques could also be enhanced through far-field EM signals (to capture EM traces far away from the emitting device) to obtain the cryptographic key.

Near-field EM signals are the only techniques illustrated in this work for this analysis, performed by the measurement test device.

2.1.1 Implementing an EM SCA attack

In cryptography, EM SCA attacks are identified as a threat to the security of information and communication systems as they make it difficult to keep the principle of confidentiality and integrity of sensitive data intact. Power scanning or EM analysis can be caused by an attacker to acquire power signals or electromagnetic emissions while performing cryptographic operations on a device. The electromagnetic samples collected by the emitting device using an EM probe over a certain period of time and at a certain rate are called traces. The collection of traces captured in one run from the scope is called capture or acquisition.

The setup to implement the EM side-channel analysis measurements, shown in Figure 2, is composed by an oscilloscope to acquire and store the EM traces, by the near field H probe which measures the magnetic field emission, by the DUA representing the system under attack, by the preamplifiers that increase the performance of the measurement device in the system and are often included in commercial EM probes showcase. Finally, the PC will synthetize and process the data acquired. Instead of the oscilloscope, for frequency analysis, a spectrum analyzer can be used.

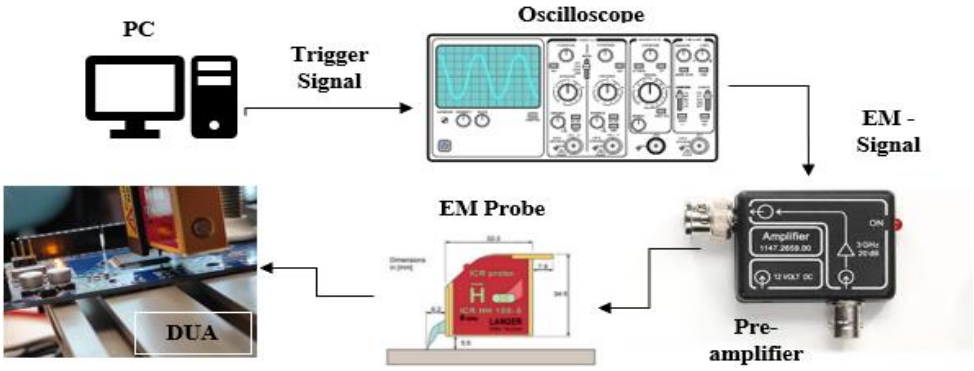


Figure 2. Ideal diagram of side-channel analysis capturing.

2.1.2 Electromagnetic probe

EM probes are used to measure nearby magnetic or electric fields. A preamplifier is located in the housing of the probe, to provide a boost to the acquired signals, and to stabilize the supply voltage [17]. The preamplifier shown in Figure 3 is placed over the

signal path between the active near-field probe and the spectrum analyzer or oscilloscope [18] [3].

EM probes are used for general purposes such as: characterizing devices, improving efficiency, and reducing emissions, and interference. In this specific work, the H probe is used to discover SCA leaks allowing the user to investigate, measure and characterize them.

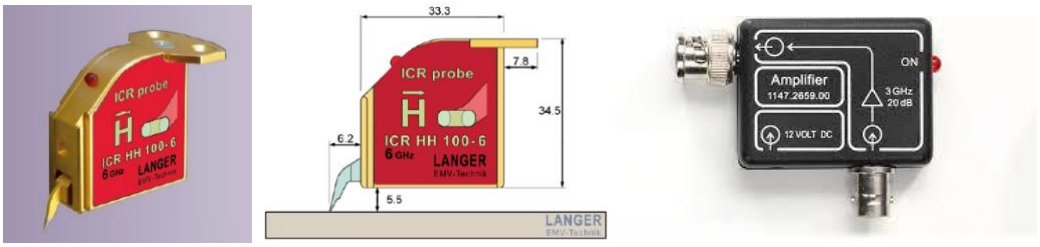


Figure 3. H probe and preamplifier.

The probe used is a commercial H probe shown in Figure 3. The Probe model is the *ICR HH 100 - 6 set*, it is a *2.5 MHz - 6GHz* near field microprobe connected to the external [19] of the EM SCA test bench shown in Figure 4. The measurement bench requires to be adjusted in order to fit the H-probe necessary to perform the acquisition of the target EM traces. The setting up of the probe in the device will be described in section 3.1.2.

Figure 4. EM Side Channel Analysis test bench.

2.1.3 The oscilloscope

The oscilloscope is an important device for EM SCA measurements. It can record multiple traces per acquisition, where the single trace is a series of samples of a signal (EM

or of power) over a certain period of time (known as a *scope window*). A trigger signal is required to begin the sampling of the trace acquisition. In general, a user can set the scope to take a defined number of samples before or after the trigger. An oscilloscope must have a wide sampling rate and a large amount of memory, to record multiple traces, where each set of traces must be separated for each trigger signal received. Such a set is called an acquisition. This ability to store and record large number of traces is the most important feature of an oscilloscope.
In theory, an oscilloscope should be able to autonomously and without interruption acquire and store a set of traces per trigger signal, requiring no manual intervention until memory is full or until the set number of triggers is executed.

Figure 5. Picoscope 3000 Series

This oscilloscope model is a suitable choice because it provides an easy implementation for the scanning system [20]. Through MATLAB, it is possible to automatically control the oscilloscope to acquire and store one acquisition per trigger signal without user interaction [21].

2.1.4 Implementing a trigger for a device

A trigger can be generated by the DUT or by a device that can help synchronize the DUT with the measurement system. This function can be provided by an oscilloscope already integrated in a main system, which supports an advanced trigger signal using different output signals from several connectors. In a real embedded system, this is not possible, the trigger is set before the entire execution of the cryptographic application.
It is possible to use connectors on embedded devices that can be used as a trigger to send specific signals to connect the DUT to the oscilloscope via USB, HDMI, GPIO, and bus cables. It is possible to generate a trigger signal for a real embedded device and academic use:

- For real embedded devices: it must be implemented as a feature by a specific signal. Through this feature, it is possible to generate trigger signals for the oscilloscope with less manipulation of the device (for example, inserting an initial acquiring command into the operating device).
- For academic purposes: it can be implemented together with the program that executes the code. It is necessary to understand the latency between the trigger in the software process and the instruction under attack.

In the thesis, the trigger chosen will be the one for academic purposes.

2.1.5 Setting up a side channel analysis test

The experiment requires that the scanning device, *i.e.,* the presented scanning bench, can locate an area of the DUT from which EM SCA traces can be acquired by running a specific algorithm designed for this purpose. The idea is to run the algorithm that performs a defined number of AES *Round 1* cycles, which are needed to characterize the `AddRoundKey` operation in the encryption device [6]. The probe implemented in the device can allow the device to perform two analyses:

1. A scan of the whole area of the device to find out the *RoA* and print the heatmap which will show the EM SCA area of the encrypting device, meanwhile the operating board will do the whole encryption at each capture set [22].
2. A scan where the probe is placed on the RoA when the system will perform a more precise analysis to define what is emitting SCA leakage and try to study the AES *Round 1* of the cryptographic algorithm. Each capture set will be an AES *Round 1 exclusive - OR* of one of the bits until all the bits of the key are processed [23].

The positioning of the probe is the critical factor of this study because it is necessary to observe the EM emissions as the probe crosses the different positions of the chip, presented as the main innovation in this work. The probe must be able to return to the same position and orientation (length, height, and width) of the point of the acquired measurements. At the same time, the measurements should ensure a precise distance at the acquisition points of 3 *μm*, because even slight changes of the probe can provide a large change in the result of the signal acquired in the traces. When acquiring SCA traces, another important piece of information to keep in mind is the device and algorithm behavior to detect when a certain operation is performed. However, embedded systems when executing an algorithm have other background processes that can alter the expected fit. For this reason, this hypothesis will be considered in two different scenarios.

- The former is an ideal scenario where it is possible to take control of the system.
- The latter is when the behavior of the embedded device cannot be controlled and therefore it is not possible to acquire knowledge of the functions of the system.

In an ideal scenario, the algorithm performing the encryption should exhibit low-amplitude SCA traces. The performed algorithm operation is not magnified in amplitude, and during the encryption, the algorithm can execute control instructions. The user must manipulate the cryptographic algorithm so that the encryption operation can be captured as SCA traces. This implies that the number of instructions has to be increased with more of the same instructions, to record high amplitude of the EM SCA traces with the oscilloscope. The algorithm manipulation implies that SCA traces presenting high amplitude are highlighted in the oscilloscope, and it should be possible to assume that SCA instructions are being acquired. A useful way for the attacker to gain knowledge about the timing of an algorithm is to know the specifications of the system. In this way, it is possible to determine the distinct phases of the algorithm under attack. Generally, the execution of the different rounds in ciphers must be divided into different loops and be acquired as different EM SCA trace sets. This requires that the user understands in which round it is necessary to operate and how to discern one round from another in an unbound execution. The thesis will consider “the first scenario”. The next section will introduce differential and correlation analysis which are methods to read the acquired data.

2.2 Definition of simple electromagnetic analysis

Simple EM analysis (SEMA) is used to analyze one trace set from the DUA. This analysis should be utilized beforehand to study the EM traces but after a detailed analysis or characterization of a device. This analysis uses the Hamming weights, which represent the numbers or bits switched from 0 to 1 visible in the scope as high amplitude Hamming weights. However, EM SCA traces amplitude related to Hamming weight travels along the bus and therefore the use of these EM traces is not as simple and straightforward as expected because embedded devices require to be modified to highlight Hamming weight or recognize those values. Moreover, Hamming weight requires to be constantly observed in the scope and pre-processed to present a peak of the previous load instruction and thus define the amplitude in all traces [24]. Traces at the same Hamming weights are amplitude averaged and it is possible to discern with amplitude alignment. Another observation that requires attention is noise, which must be adjusted to infer the Hamming weight from a single trace. SEMA if used in conjunction of Simple Power Analysis can be useful for determining algorithmic structure as opposed of Hamming weight [25].

SEMA can be used to extract the secret key if the key-dependent operations are performed and discovered. Anyway, SEMA is a difficult operation to perform since traces look like random noise [23].

For this purpose, it is necessary to operate with more advanced analysis as shown in Figure 6, neural networks can be used to improve this process [26].

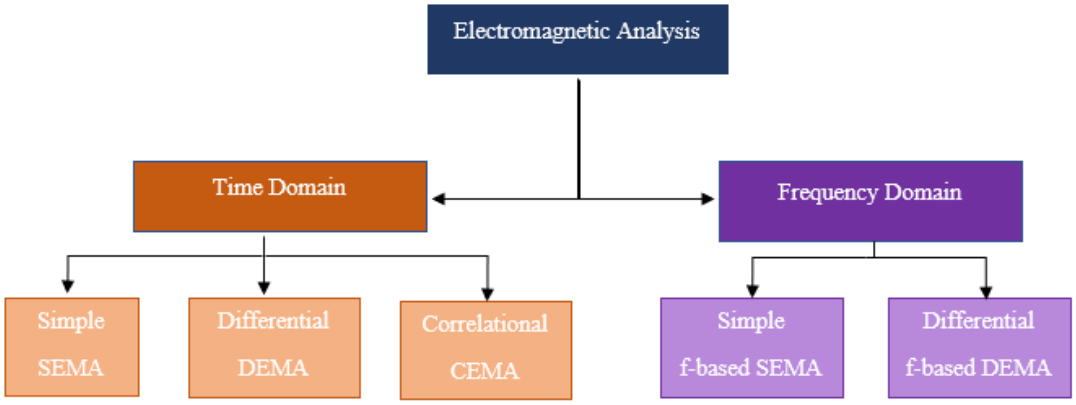


Figure 6. Electromagnetic analysis evolution.

2.3 Implementation of differential analysis

Differential analysis can be used to look at the common operational part which are the operations used by algorithms to encrypt and decrypt information in many cryptographic algorithms. In this thesis, a DEMA was used. Block ciphers running on 32 bits processors offer a fast operational implementation. As shown in Figure 7, the input is plaintext, and the key is involved in many calculations that will yield an indexed table (S-box table). Two assumptions need to be made:

1. Assume that the plaintext and the key are combined with *exclusive-OR* and then collected in an index table. In this case, the AES is the cryptographic algorithm used.
2. Assumes a priori clues, the plaintext values, and the S-box table values (because for Kerckhoffs's principle [27], the encryption algorithms are published).

The attacker begins his attack maneuver with two ideal attack practices that are shown in Figure 7, the first at the output of the *exclusive-OR-ed* operational, the second at the output of the S-Box [2]. In general attacks, that operate at the operational logical part of an embedded processor requires that the chip has been decapsulated and that a probe is placed very close to the right emitting part, to perform near field EM SCA. Another alternative may be that the output of the exclusive or operational part is correlated with the input to the S-box memory, thus correlated with an attack to the address used to access the S-box memory [28].

But the attack needs to be added that the specific memory addresses used by a processor cannot share the same address values for the S-box (0-255), hence they can be enabled to conduct an attack.

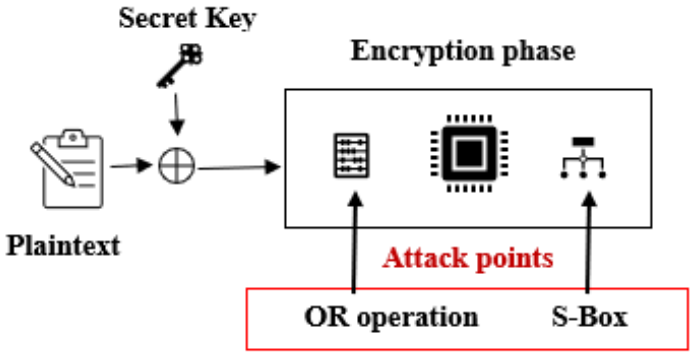


Figure 7. Electromagnetic analysis evolution.

"This kind of attack is more difficult on 32-bit medium size processor because the address width in an 8-bit input S-box is less than the data access width from the memory in a 32-bit out from S-box" [29]. *Gebotys 2006*. Hence, if the S-box is not inside the chip but in an easy targeting position, there is a greater likelihood that the attack will be successful. This effect is caused by a large amount of power consumption or dissipation of energy from the operation as external access, these massive emissions surely will present a detectable side channel, and thus a greater chance of successful attack. Moreover, memory and bus integrated into the chip present a more complex switching operation to attack in comparison with the off-chip data transfer. Therefore, the side channel signal strength relies more upon the underlying embedded system. However, 8-bit processors and memory are completely integrated into one chip, with many implemented countermeasures. In this way, the registers and the logical and arithmetic operations are protected because they have a low power dissipation and therefore the embedded processors are more difficult to attack. To perform this kind of attack strategy, an easier way is to use a decapsulated chip with a probe positioned on the silicon. In other words, differential analysis tries to expose the logic register switching, by positioning a probe on the silicon area of interest, which is different from an embedded cryptographic platform where all the activity is gathered on the cryptographic core.

Now, a mathematical procedure will be shown to execute a differential attack delivered to the input and output of an S-Box. This kind of attack records the first EM or current measured from the device while the processor executes the cryptographic algorithm.

The representation of *n* traces from $C_0, C_1; \vdots\vdots\vdots: C_{255}, C_{256}; \vdots\vdots\vdots\ C_{n-1}$

Which are generated from *n* plaintext $p_0, p_1; \vdots\vdots\vdots\ p_{255}, p_{256}; \vdots\vdots\vdots\ p_{n-1}$.

Note that $p_i = i\ for\ i = 0; \vdots\vdots\vdots; 255, and\ p_{256} = 0, p_{257} = 1, etc: \vdots\vdots, or\ p_i = i\ mod\ 256.$

Several traces are acquired from each plaintext input to the secret algorithm.

Formulas 2.1 and 2.2 are the SCA values used to refer to EM SCA traces.

$$p_i = i^{th}\, plaintext$$

$C_i(t)$ = sample at the time t of i^{th} side channel trace corresponding to the i^{th} of the plaintext$\{X\}_j = \ the\ j^{th}\ bit\ of\ X\ (e.g., X = (01011011), the\ 0^{th}\ bit\ is\ 1, the\ 1^{th}\ bit\ is\ 1, \dots)$

K = key guess (which may value 0 to 255 for an 8-bit key)

$$A_i^K = \begin{cases} p_i \oplus K, & at\ input\ of\ S\text{-}box \\ S(p_i \oplus K) at\ outpiut\ of\ S\text{-}box \end{cases}$$

$D^{j,K}(t)$ = differential trace based upon partitioning with the j^{th} bit of attack point A_i^K for key guess *K*.

$D_S^{j,K}(t)$ = the significant differential trace based upon partitioning with the j^{th} bit of attack point A_i^K for key guess *K*.

n_b= number of traces whose plaintext p_i obey $\{A_i^K\}_j = b$

(Note that it is also a function of *(j, K)* however for simplicity, it will not show the super-script *(j, K)*

Where $b = \{0,1\}$

The Differential Trace

$$\mu_b^{j,K}(t) = \frac{1}{n_b} \sum_{i \mid \{A_i^K\}_j = b} C_i(t),$$

$$D^{j,K}(t) = \mu_0^{j,K}(t) - \mu_1^{j,K}(t) \qquad (2.1)$$

The Significant Differential Trace

$$D_S^{j,K}(t) = \begin{cases} D^{j,K}(t) - 2s, if\ D^{j,K}(t) > 0 \\ D^{j,K}(t) + 2s, if\ D^{j,K}(t) < 0 \end{cases}$$

Where,

$$\sigma_b^{j,K}(t) = \sqrt{\frac{1}{n_b} \sum_{i \mid \{A_i^K\}_j = b} \left(C_i(t) - \mu_b^{j,K}(t)\right)^2}$$

$$s = \sqrt{\left(\frac{1}{n_0}\right)\left(\sigma_0^{j,K}(t)\right)^2 + \left(\frac{1}{n_1}\right)\left(\sigma_1^{j,K}(t)\right)^2} \qquad (2.2)$$

The differential trace and significant differential trace equations defined in (2.1) and (2.2) show the two attack points defined in Figure 7. Means and difference of means can be used to remove algorithmic noise. The value $\sigma_b^{j,K}$ is the differential trace, and the *s* is the significant differential trace, and it is the standard deviation of the differential means. These two equations represent the two attachment points represented in Figure 7. The means are collected to reduce the noise value, while the difference of means is used to reduce the algorithm noise. Data dependent variation is not removed from the differential

traces [29]. The significant differential traces *s* is the standard deviation of the difference of means. *"Thus, the s value is the combination of all the differential traces that are left outside the plus and minus of the two standard deviations of the difference of means". Gebotys 2006.*

These two values are statistically significant, they can be adjusted for different kinds of ciphers where an attack point is defined as a general function of: key guess (K), partitioning bit *(j)* and plaintext value *(i).*

The procedure to launch a side channel attack at the S-box input begins with the assumption that the traces containing the attack point have already been acquired.

Procedure 2.1. Attack at Input of S-*box*

1) Input: $C_i(t)$, p_i, where $c_i(t)$ is sample a time *t,* plaintext p_i is a plaintext that goes from $i = 0;:::; 255.$
2) Output: k_j; $j\ D\ 0;:::; 30$(secret key value)
   Set $K = (0000000)$
   For $j = 0\ to\ 7$
   a) Compute $D_S^{j,K}(t)$ or $D^{j,K}(t)$ Significant Differential Trace or Differential Trace
   b) If the plaintext load spike is in the same direction as the S-box load spike, then $kj = 0$ else $kj = 1.$

The differential trace should present two traces as shown in Figure 8. The first spike (from left to right increasing time direction) corresponds to the plaintext load, meanwhile, the second spike corresponds to the S-box load. *Procedure 2.1 Line 2b* requires that the attacker will determine the direction of the second spike relative to the first spike, where the height of the spikes will represent if the bit was a one or a zero $k_j = 1$. The single load will produce one positive spike and one negative spike with many oscillations in between.

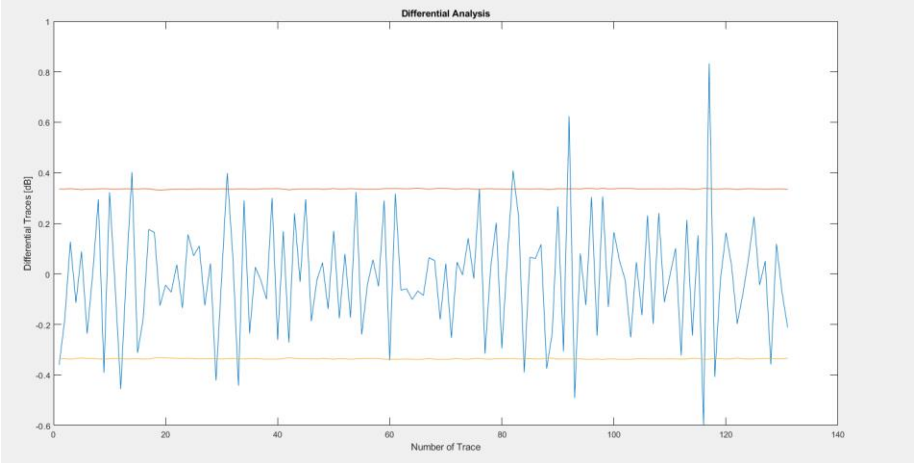


Figure 8. Differential analysis plot

The majority of the differential traces will be acquired as a combination of extreme positive and negative peaks. The two standard deviations will present differential traces, at first high and second low. In the second graph (Figure 9Figure 6) the situation is the opposite, at first low and second high, caused by the key bit effect.

The least significant bit could be used but needs to be analyzed because it changes from processor to processor.

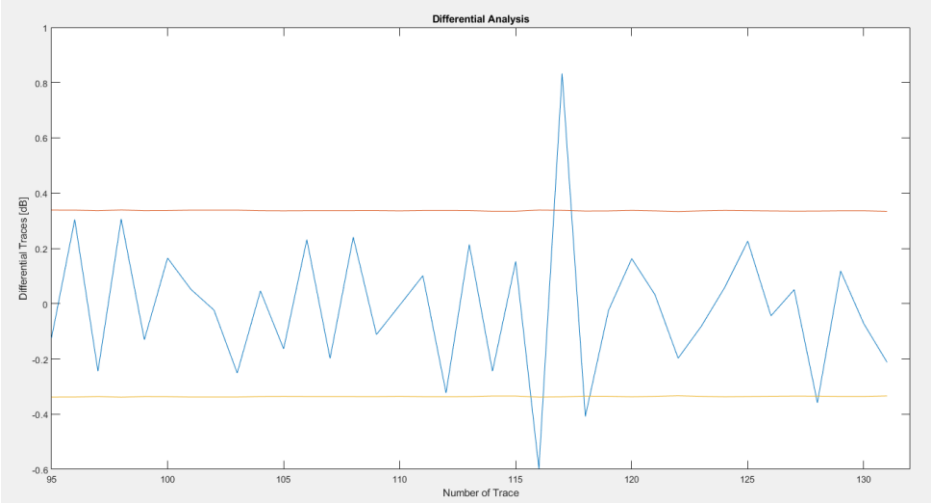


Figure 9. Differential analysis peak of interest

The value H_K is the Hamming weight of the predicted key value, that has been measured with the differential analysis procedure.

$$H_K = ma\,x_t \left| D_S^{j,k}(t) \right| \tag{2.3}$$

Procedure 2.2. Attack at Output of S-box

Input: $C_i(t), p_i$

Output: K (8-bit secret key value)

1) Select a partitioning bit, j , $0 \leq j \leq 7$
2) For $K = 0\ to\ 255$
   a) Compute $D_S^{j,K}(t)$
   b) Compute H_K
3) $K = \{K \mid H_K > H_i\ for\ all\ i \neq K\}$

Procedure 2.2 “Attack at the output” of S-box provides one key byte, but this procedure needs to be executed for all the 16 bytes of the plaintext (16 bytes for every of the 16 S-box accesses). Obtaining the entire *128-bit key* H_K requires it to be computed as the maximum absolute value of the significant differential trace which represents the *“height of the differential* (H_K)*”.* The plot in Figure 8 and Figure 9 shows the H_K/K value as an all-keys plot. By this, it is possible to determine the correct secret key.

A differential trace partitioned with a correct key in AES with the relative average traces, measured with the measurement device's configuration is shown in Figure 8. The two standard deviations are plotted above and below the zero axis. The sections of differential traces, that increase and decrease exceeding the limits of the standard deviation, indicate a high enough probability to find a leakage in the SCA. These peaks are identified in correspondence with the S-box output. Figure 9 presents the graph of the most magnified peak of the differential height regarding the key. If the operation with the key presents a very similar signal height, the attack is more difficult, and more traces are necessary.

The number of traces for the correct key hypothesis in correspondence of the S-box output with AES is the amount reported as the axis of the differential analysis plot. In both plot cases the *"bit one"* which is the less significative was used to separate the traces EM of the first order.

To create these two plots, two different acquisitions are being used with the same collection of data. The EM noise is the main cause of the difference in the results presented. The two standard deviations decrease with the increase of the number of traces and the DEMA height is reducing too. The necessary number of traces can be 100 EM traces for each partition. With 200 EM traces, it is possible to obtain a DEMA peak of interest, as shown in DEMA Figure 8.

Multi-bit differential analysis can be performed by summing (*Σ*) the differential heights H_K for a key guess K over all different partitioning bits.

2.4 Correlation Analysis

Correlation analysis is another technique that relies on the correlation of the EM power measurements in combination with the Hamming weight of the predicted data. It is more powerful and more recent in comparison with differential analysis.

Correlation Analysis symbols are:

n = the number of traces

$p_i = i^{th}$= plaintext

K = Key guess

W_i^K is the Hamming weight of the predicted S-box output for the i^{th} plaintext based on key guess *K.*

$C_i(t)$ = sample at time *t* of the i_{th} trace generated using plaintext *i.*

$\rho^K(t)$ = Correlation factor at time *t* based on the key guess *K.*

The Hamming weight is measured: $W_i^K = HW(S(p_i \oplus K))$

Where a more precise version is $W_i^K = \frac{cov(C,W)}{(\sigma_C(t)\ \sigma_W^K)}$ which result

in $W_i^K = \frac{E_{C,W}^K(t) - \mu_C\,(t)\,\mu_W^K}{(\sigma_C(t)\ \sigma_W^K)}$; (2.4)

The operators of the previous equation are:

$$E_{C,W}^K(t) = \frac{1}{n}\sum_i C_i(t) W_i^K$$

$$\mu_C\,(t) = \frac{1}{n}\sum_i C_i(t),\ \mu_W^K = \frac{1}{n}\sum_i W_i^K \quad (2.5)$$

$$\sigma_C(t) = \sqrt{\frac{1}{n}\sum_i \left(C_i(t) - \mu_C\,(t)\right)^2}$$

$$\sigma_W^K = \sqrt{\frac{1}{n}\sum_i \left(W_i^K - \mu_W^K\right)^2}$$

$$H_K = \max_t \{|\rho^K(t)|\}$$

To identify a guessing key *(K),* and so forecast the output values of the S-box for each plaintext, it is possible to utilize a procedure of attack at the S-box output. To correlate every sampling point in a defined trace it is necessary to utilize the Hamming weight of the predicted value.

Every single point in the specific trace is correlated by the Hamming weight of the predicted value defined as a correlation trace. It can be plotted as a correlation factor versus sample, regulated by the W_i^K equation as shown in Figure 10.

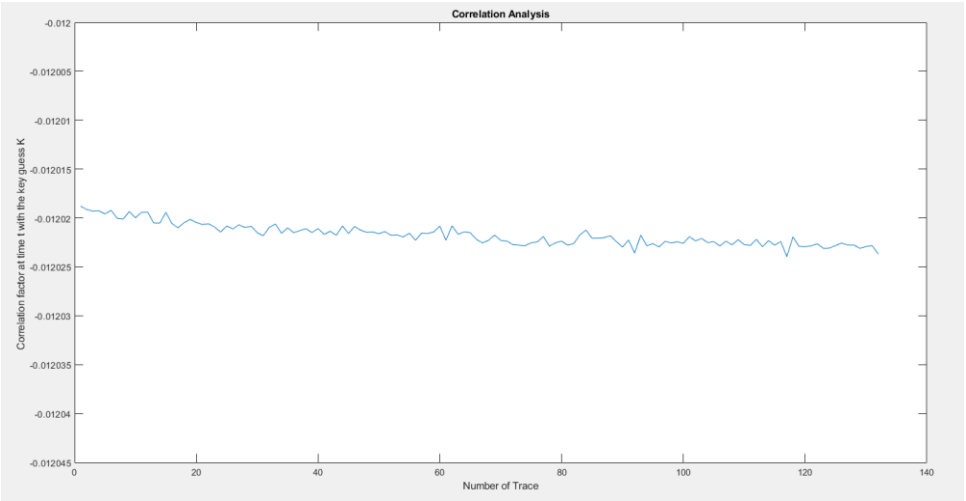


Figure 10. Correlation analysis

Usually, the corresponding EM trace avoids showing the shape of the EM trace to protect the technology.

2.5 Differential Frequency-based Analysis

Differential frequency-based analysis is an extension of the differential analysis, where, instead of computing the differential signals in the time domain, the computation is done in the frequency domain as differential frequency analysis (DFA) [29]. DFA helps in the resolution of the misalignment trace because the Fast Fourier transform (FFT) provides time-shift invariance [9]. Frequency analysis is a necessary way to improve the temporal analysis, not only to avoid trace misalignment but also to help to avoid loops and repeating structures in the algorithm.

Differential frequency analysis is described as follow:

$C_i(t)$ is the trace corresponding to plaintext p_i where *i* is the trace number

$i = \{0, \ldots., n-1\}$,

K is the key guess,

$C_i^b(t)$ is the EM signal of the set b $C_i(t)$ where $\{S(p_i \oplus K)\}_j = b$; where $b \in \{0\ or\ 1\}$,

$FFT()$ is the fast Fourier transform.

$$P_i^b(f) = \frac{\left|FFT\left(C_i^b(t)\right)\right|^2}{n}$$

$$D_j^K(f) = \frac{1}{\frac{n}{2}}\sum_i P_i^0(f) - \frac{1}{\frac{n}{2}}\sum_i P_i^1(f) \tag{2.6}$$

$$H_j^K = \sum_f D_j^K(f) \tag{2.7}$$

The key plots are obtained by summing (Σ) the difference in amplitudes over all frequencies using equation H_j^K. The standard deviation could be subtracted with the difference of means from the differential signals in the frequency domain as shown in equation (2.6) as shown in Figure 11.

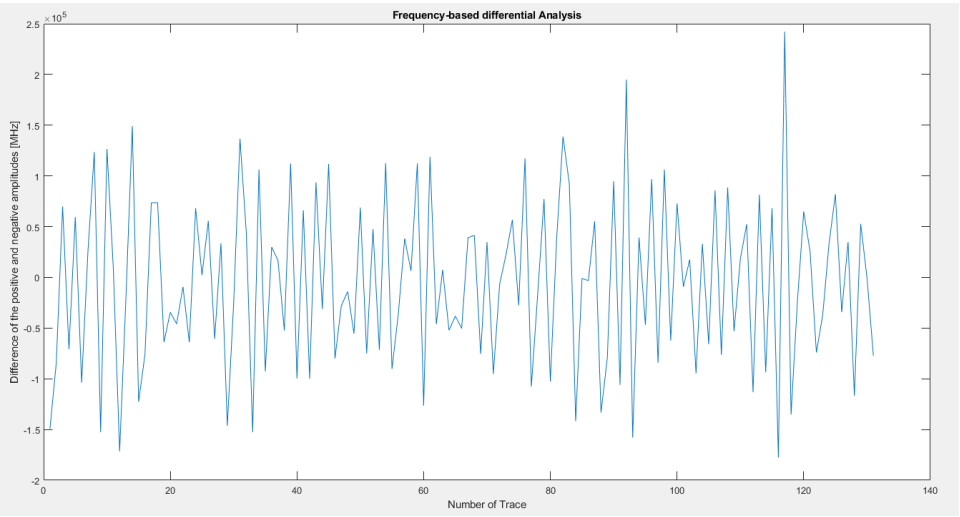


Figure 11. Differential frequency analysis

The normal way to operate should be to first apply differential frequency analysis (DFA) on the different parts of the trace. This way it is feasible to locate the possible areas of attack and next to focus on the smaller areas of interest. Normally, this procedure is performed by acquiring the EM traces at the central part of the *RoA*. Hence, it is possible to study random selection and determine the time shift or delays. *Frequency-based DEMA* on a specific area defined as *RoA*, with multiple sets of acquired EM traces using different keys, may help to characterize the device and so obtain the secret key of the device. This process should be repeated for each key byte of the key, until the 128-bit encryption key is found.

2.6 Performing a lunchtime attack on Personal Computer Device (PCD)

A PCD could be a cellphone, a clock, a memo-pad, or a calculator. An attack may begin not only from device theft or loss but also from the daily unintentional electromagnetic waves radiated from the wireless device during cryptographic computations, which can leak confidential data to a nearby attacker. An example would be the procedure of a lunchtime attack to uncover the secret key stored in confidential memory in a wireless device. This attack can be a result of a loss or theft of a device, if the device is modified to facilitate intercepting the EM waves being emitted while performing cryptographic computation, making future wireless communication insecure [30].

A methodology for EM attack on an embedded system is called device characterization. It works as if the device is under attack because the attacker can execute and modify the same encryption algorithm implemented in Java on the characterization device. The difficulty in a real embedded device is misaligned trace due to system interruption, poor trigger, and other circumstances. But the extraction of a key is possible even without perfect alignment, by *frequency-based DEMA*.

The characterization of a device can be described in a flowchart which describes the different procedures and necessary steps to create one [28].

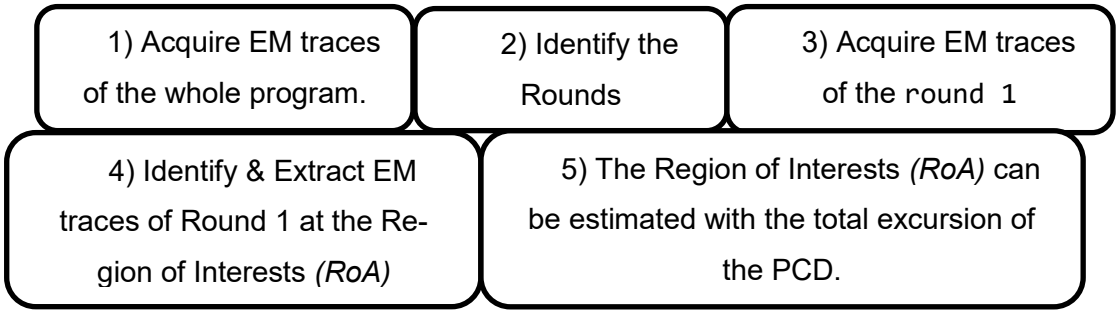


Figure 12. Lunch time attack procedure

As shown in Figure 12, these steps are a way to proceed to collect the EM signals from the characterization device.

1. During the execution of the cryptographic algorithm with the help of a SEMA it should be possible to identify the rounds of the encryption algorithm.
2. It is possible to begin to collect the traces of `round 1` from the encryption algorithm when the S-box outputs are accessed in AES encryption. The characterization of a trace-to-trace misalignments associated with the acquisition of traces requires several EM acquisitions in the region of attack *(RoA)* to estimate the time shifts.
3. Performing a final acquisition to achieve the proper resolution and sampling of most of the traces emitted by the *RoA*, *the trace width is the RoA width plus two times the average trace-to-trace delay typically.*
4. With the help of a pattern recognition algorithm, the EM traces are extracted. The recognition algorithm should be able to discern between thresholding, regions of activity, and EM magnitude. In this way, it is possible to determine a set of rules to extract the *RoA* information.
5. Examining different areas of the DUA in the *RoA* can be useful to understand if the RoA is divided into more leaking areas of the device.
6. Those areas are extracted and analyzed with frequency-based DEMA.
7. Key checks for the correctness and characterization of the device are completed and from now on the *RoA* will be defined as $Area_a$.

A complete and working DEMA requires several sets of acquisitions to discover the proper *RoAs* and so to obtain the key and characterize the device and create a methodology of attack.

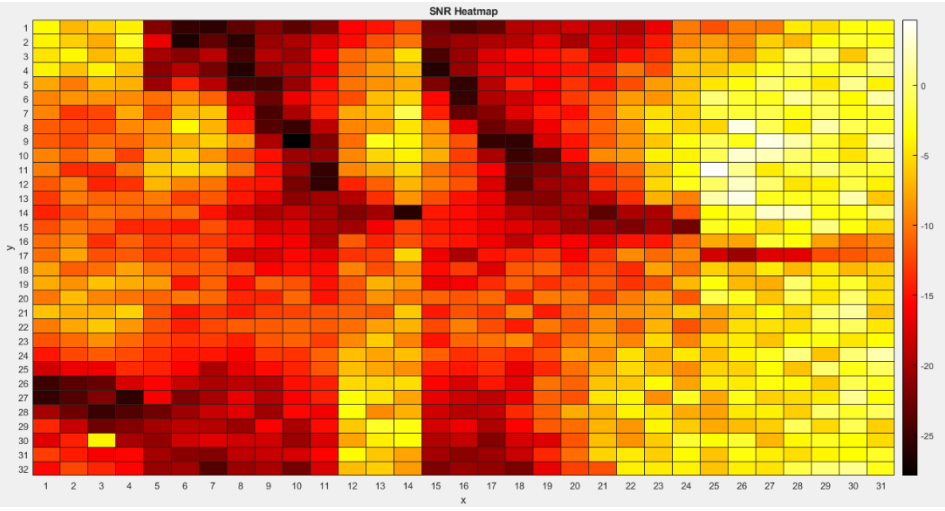


Figure 13. Heatmap to define the Region of Attack (RoA).

The SEMA needs to find and acquire a certain *RoA* performing a multitrace acquisition as well as to identify the right area of acquisition called $Area_a$. Figure 13 shows a heatmap, where it is possible to locate a few *RoA*. $Area_a$ in this case is in the central part of the map. In the last part, DEMA is applied to discover the device key. Usually, this process needs to be repeated for each key byte *(k)* within round 1 until the complete 128 – bit encryption key is found. DEMA alone is not sufficient to discover the key because of the perfect trace alignment, so it is necessary to use an *f-based DEMA* which should avoid the misalignment.

3. MATERIALS AND METHODS

This section will describe the assembly and implementation of the acquiring bench, the procedures to acquire and process EM SCA traces measurement, followed by the procedure of how to carry out an EM SCA against the cryptographic algorithm AES. The implementation of the engines for the three coordinates of the measurement bench will be shown with the pin configuration through the Arduino IDE, that constitutes one part of the necessary software to interface the hardware to the measuring device. The software will be used as a demonstrative example to present the measurement bench tools, in order to show the operation of connection to the test device, heat control, and then the autonomous software to characterize the DUT and define an *RoA* implemented as `Test 1,` while `Test 2` allows to the user to capture the EM traces in the maximum leakage point discovered, followed by the script to perform *DEMA, CEMA, and f-DEMA* [31]. The implementation of the hardware control through serial port by MATLAB is necessary to monitor the bench, the oscilloscope, and the different software simultaneously. Then, the system to record the EM SCA leakages is organized through a *picoscope* (oscilloscope), by the MATLAB [32] [21] software implemented in the system with the function called `Picocapture`. Moreover, the cryptographic algorithm (AES) will run through the operational board (Arduino 2560) to simulate PCD, in this way, it will be possible to study how to define a *RoA* for the attack.

The manipulation of the PCD is used to loop *round 1* of encryption with Arduino IDE and the decision made to acquire the measurement with the related delay case study will be discussed to provide a better understanding between device trigger and acquisition delay. Those features are essential to organize the scanning device efficiently.

To process EM SCA measurements software is necessary to interface the acquiring system and the measuring system, to execute the needed operations of movement, and to acquire the traces autonomously [33]. The user needs only to define a region of interest through the implemented commands in the system and to analyze the result obtained results through the RoA study case.

3.1 Side channel analysis measuring platform.

The EM side-channel attack allows an attacker to choose the location with the maximum information leakage, which is calculated through the SNR, introducing a series of additional demands. To carry out this procedure it is necessary to reduce the amplitude of

the signal concerning the measurement of the background noise (noise floor), which results in more traces or more expensive measurement equipment needed to perform EM SCA attacks. In addition, unlike power attacks, EM SCA attacks require attackers to choose the location of leakage points on the device to capture EM SCA traces. This choice can impact the effectiveness of an EM SCA attack depending on where the probe is positioned on the chip. To determine the position of maximum leakage to perform the SCA attack, it is necessary to perform an exhaustive search in most of the points of the cryptographic device area, considering that, being the surface is exceptionally large, finding the correct location could be difficult because it would require a scan of the entire chip or system. The system proposed in this work uses a measurement bench as an EM SCA test bench that will allow the SEMA to perform the measurements to identify the spatial points with higher SNR (POIs). The device performs two different analyses: the first is a gradient heuristic search using a dispersion measurement, which records test vector leakage assessment (TVLA) and SNR values to locate the point of high leakage quickly and automatically on PCD [8]. Once the point at high leakage has been determined, it is possible to use a second analysis on a PCD, such as correlational or differential EM SCA analysis (CEMA / DEMA). [33] Both CEMA and DEMA are possible EM SCA attacks that, when used in this way, may increase the risk of an IoT device to be compromised [35].

The side channel system consists of two parts implemented as `Test 1`:

1. An automated system is used to scan the cryptographic device, to find secret data-dependent leakage points, by means of the EM SCA test bench implemented on the measurement bench.
2. A gradient heuristic search algorithm that converges at a point of high loss of the cryptographic device, evaluating TVLA or SNR as a measure of loss.

However, making these EM SCA attacks practical and real-time on embedded devices requires automating the process of collecting EM SCA traces. Losses are measured with TVLA to SNR, more precisely two sets of traces are collected. The first set contains key, and plaintext is used as input to the algorithm under test and kept as a fixed sample; the second set with randomly varied plaintext but encrypted with the same fixed key. [36] [5] Once the EM test bench has established the point of EM SCA attack, these points can be used to perform EM SCA attacks such as DEMA and CEMA [25].

3.1.1 Automated test bench for SCA scanning

Here is presented the description of the physical components needed to build the device together with the implementation of the automation of the system. The scanning hardware consists of a bench which can move in the three coordinates (X, Y, and Z) with an EM field probe and the Arduino Mega 2560 platform that allows the interfacing of the X and Y motors and the movement related to these two coordinates.

Figure 14. Arduino Mega 2560 board

Figure 14 shows the Arduino Mega 2560, a microprocessor that allows the user to interface a laptop with the measurement bench and launch the scanning program.
The pin configuration will be shown in the algorithm section.
The test bench can be controlled via the Arduino Console Serial Monitor. The speed and acceleration are $1000\ (step*s/mm)$ and $1000\ (step*s^2/mm)$, these numbers are discussed in the algorithm part. The *H-field probe* scans the area, while the oscilloscope stores the signal captured and records the EM traces.
The H probe has an extremely high spatial resolution, it can capture leakage of information from the device when it is in the appropriate position [17] [37] [38]. Even if the probe is oriented towards the larger area surface, with the knowledge of the DUT it is possible to enhance the system efficiency, reducing the required time for scan. The position of the probe can be controlled from the Arduino serial terminal and MATLAB serial port script as shown in Figure 15.

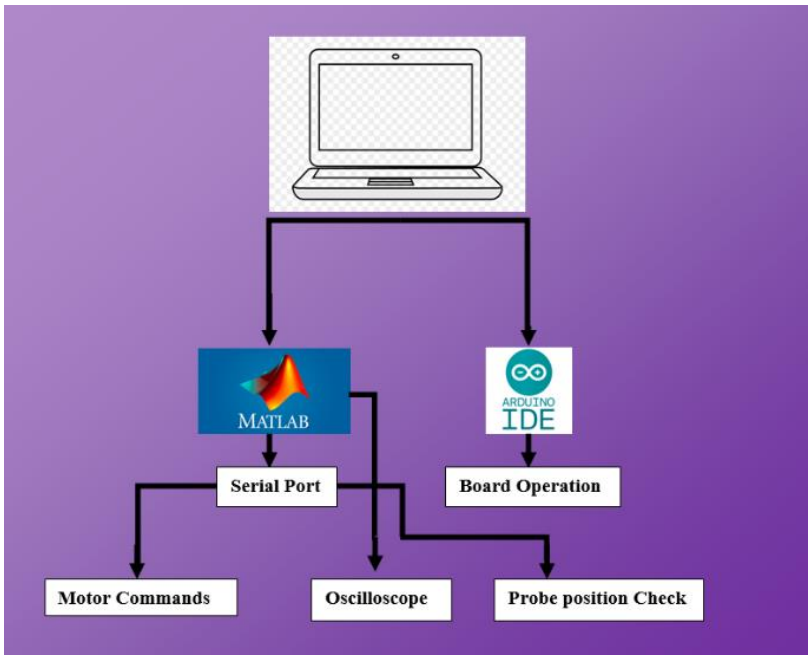


Figure 15. Motor diagram to control the measuring system.

Figure 16 shows how motors are controlled by a microcontroller, in our case three Nema 17 motors are used, two for controlling the X and Y position and the third to position the EM probe along the Z coordinate [39].

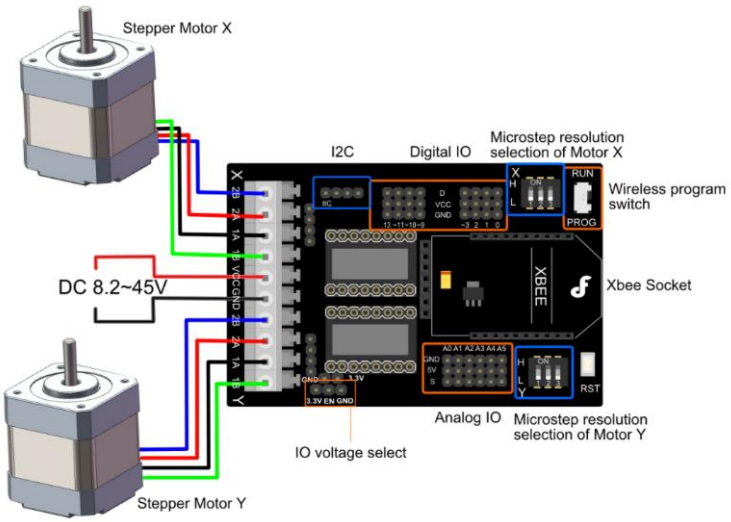
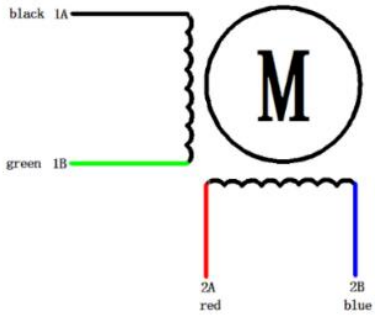


Figure 16. Motor diagram with pins configuration [39].

The motors for surface scanning follow the table in Figure 16, which describes the connection of the pins to control the coils.

In total, there will be 2 pins for direction and time control which will describe the coil activation interval for all three motors.



MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	1/4 step
High	High	Low	1/8 step
Low	Low	High	1/16 step
High	Low	High	1/32 step
Low	High	High	1/32 step
High	High	High	1/32 step

Figure 17. Engine function tables [39].

The coil diagram is shown in Figure 17. The pin configuration must be declared in the code to allow it to work. On the right, the resolution of the step which in the case used corresponds to the fifth line $Low,\ Low,\ High$ with precision $\frac{1}{16}$ of a step, which should ensure the required accuracy to be able to scan all the microcells of the device under examination.

Figure 18 shows the two motors for scanning the surface and the raised section where the third motor is located with the probe for depth control and trace acquisition. After describing the low-cost hardware of the scanning system, it is now possible to move on to the EM SCA phase and then to the procedure for automatically scanning the DUT.

Figure 18. Measurement bench engines.

As announced before, the basic premise of the automated system is to locate a point on the target device where there is a chosen high leakage using the algorithm, then run SNR and TVLA at those POIs. This step excludes the need for an expert to manually trace the attachment position. During this phase, the probe is in the desired position through the scanning algorithm, for the trace collection phase.
The signal first passes from an external amplifier and then to the internal amplifier of an oscilloscope, to acquire the leakage via SNR or TVLA in the position of highest leakage as shown in Figure 19.

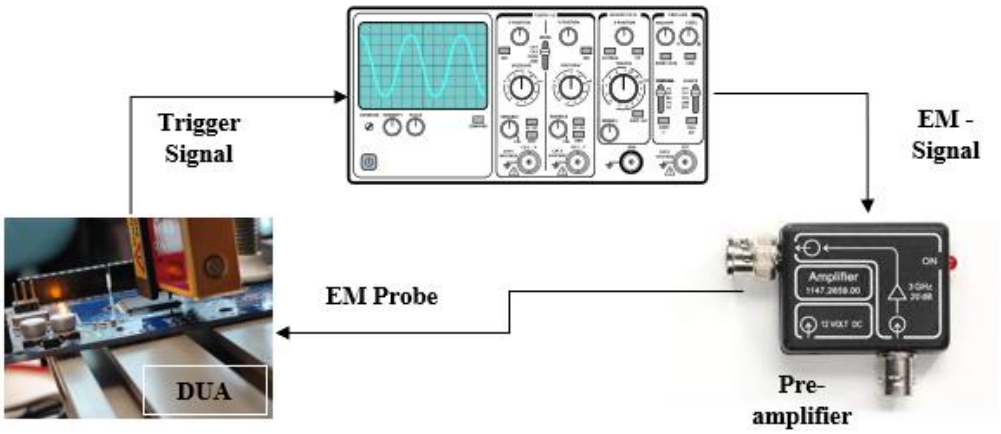


Figure 19. Measurement procedure.

3.1.2 Assembling the probe support

A critical phase of the system's construction has been the assembly of the probe support produced with a 3D printer. The support shown in Figure 20 and composed of the three segments shown in Figure 21, was designed in a way that it was possible to insert inside the XYZ bench turret and so allows the connection of the probe to the bench as shown in Figure 19.

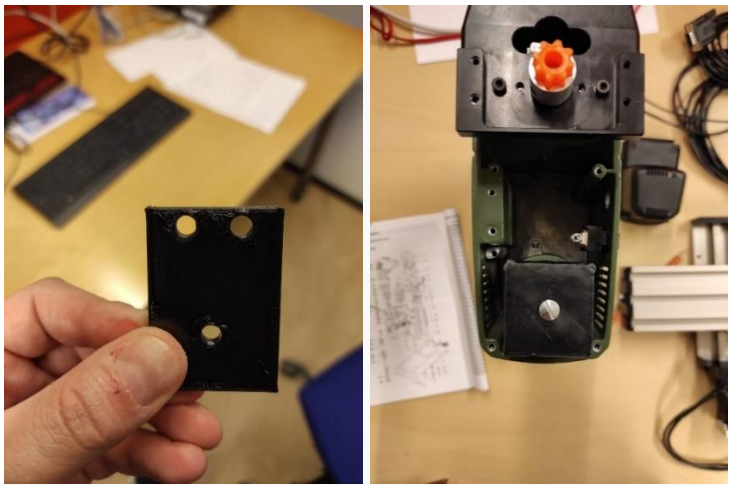


Figure 20. Probe support

These three components' *first, second* and *third segments* are positioned to hold the probe under the Z turret.

Inside the box of the turret, a combination of the *second segment* and *third segment* was necessary to hold the main central screw, in a way that the probe was still and well fixed.

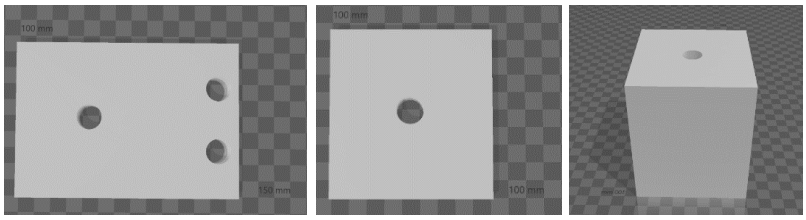


Figure 21. Probe supports segments.

The first segment has a length of 5.3 cm, height of 3.7 cm, depth of 0.4 cm, and a hole radius of 0.3 cm, the second segment has a length of 3.7 cm, the hole has a radius of 0.3 cm, the third segment has a length of 3.7 cm, but depth 4.2 cm.

3.1.3 Loss measurement by test bench for EM SCA scanning

The choice of EM probe location is the critical factor for determining the traces required to retrieve the encryption key via EM SCA. Currently, this process is performed through an exhaustive search of the entire chip, evaluating the EM emissions of the distinct positions of the system. The operative search will report the position of greatest loss which will be identified as the best point to carry out the EM SCA attack. Unfortunately, it will take a certain amount of time because the measurements reported will be numerous, especially if the system has a great initial disclosure value of the minimum traces MTD (Measurements to Disclosure). It should be added that the choice of the traces position based on the visual inspection, can be effective, but not necessarily better in terms of MTD. In this work, a method will be developed that completely automates the process of selecting a position with a high loss without the need for an expert to carry out the measurements for the detection of the correct position. This could be the point where the MTD is lowest, but it gives enough information to implement an EM SCA attack in a reasonable amount of time and without the need for an expert. The test bench is designed so that it can exploit several types of losses such as signal amplitude, Test Vector Leakage Assessment (TVLA) [40], or the Signal-to-Noise Ratio (SNR) [23] as a measure of the loss, the platform will carry out the measurements of the loss at the designated point.

- **Signal Amplitude.**

  Signal amplitude measurements are performed at each point of the chip under examination. The amplitude is measured as the mean square amplitude of each

trace, average for every 10 samples. The signal amplitude is quick to measure but has no relation to the loss of the side channel.

- **TVLA**

  They are measured in high loss values with good accuracy and without including a low MTD value. The TVLA scans enough points in each group. This number of traces generates unambiguous evidence between the points of low and high dispersion. However, it is not possible to take many measurements at each point to calculate the average of this noise.

- **SNR**

  The operation to calculate the SNR allows one to reach the maximum SNR value by minimizing the MTD values. SNR is calculated using the same CEMA loss model or the First Round S-Box Output together with the Hamming Weight Model.

The loss measurements performed with the three methods are all valid alternatives to be used for an EM SCA since the goal of the system is to discover the weakness of a device and understand how these measurements are related to the MTD value in the distinct positions of the DUT [6] [23]. It is possible to scan the surface of the DUT and subsequently collect 1000 traces for each point. In this way, it is possible to perform the TVLA at each point. With this method, SNR and TVLA are correlated, and it is possible to analyze the points where the MTD [15] [41] is present in greater quantity, even if the correlation with the amplitude is always low. It must be added that the signal amplitude is easy to detect, but the MTD measurement is not available, so due to the high signal losses there is no certainty of being able to intercept the information [30]. Therefore, unrelated EM sources with high signal loss could confuse the attacker and lead him to choose a weak attack point [42]. While TVLA does not guarantee a high loss value, it can still be used to identify regions where adequate losses are detected, also ensuring adequate MTD in this process [43]. It should be added that SNR is a necessary tool for EM SCA because it has a remarkably high success rate and defines the high SNR location with the lowest MTD value, which is the goal of the measurement bench [22].

3.2 Algorithm

In this section the algorithm for locating the point of leakage where the EM SCA attack should be performed is presented. Through the algorithm, the system can exploit the weaknesses of a device to search for the point of greatest leakage. To avoid making measurements at every spatial point, a search algorithm is proposed to find a high leakage point with a minimum number of scans. The search algorithm works in two tests:

- `Test 1` scans an area defined as the movement along a surface.
- `Test 2` scans at a predetermined point.

Before deepening the test codes, the first part of the algorithm will be shown, beginning with the definition of the pins connected to the Arduino microprocessor shown in Program 1 [44] [45].

```
//Libraries:
#include <AccelStepper.h>
// Define Pin Connections
int M1dirpin = 7;          // Motor X direction pin
int M1steppin = 6;         // Motor X step pin (clock)
int M1en=8;                // Motor X enable pin
int M2dirpin = 4;          // Motor Y direction pin
int M2steppin = 5;         // Motor Y step pin (clock)
int M2en=12;               // Motor Y enable pin
```

Program 1. Connections for pin

The Table 1 shows the commands implemented in the Arduino microcontroller by inputting `H (help)` in the operating console. These commands can be used through the MATLAB serial port to execute the `writeline` script [46].

Command Implemented	Letter
Help	`H`
Quit	`q`
Simple movements	`wads`
Standby	`S`
Trigger	`t`

Table 1. Commands set for device control.

The code shown in Program 2 uses library `AccelStepper` [47] performs the standby phase entered if the user presses the `S` key. It is used for energy saving and to cool down the motor during breaks without the need to switch off the engines.

```
while(val == 'S'){
  rec = digitalRead(inpin1);
  myStepper1.enableOutputs();       // disable outputs
  myStepper2.enableOutputs();       // disable outputs
   if(j<1){
     Serial.println("Standby");
     j++;}
     digitalWrite(outpin1,HIGH);
     delay(100);
     val = Serial.read();
   }
 }
```

Program 2. Standby

Program 3 shows the basic movements of the motor using the *"a"* command, allowing the motor to move in the left direction.

```
// Simple Movements
if (val == 'a'){
  j = 0;
  delayMicroseconds(2);
  for(j=0;j<=1000;j++){
    myStepper1.setSpeed(2000);
    myStepper1.setAcceleration(1000);
    myStepper1.move(+2400);// move 1000 Clockwise (Positive-Down)
    myStepper1.run();
    delay(1);         }
              }
```

Program 3. Simple movements on left and right.

The remaining movements of the motor are right, up, and down simply by simply pressing *"d", "w",* or *"s"* key. The system has been designed to be very simple, to reduce problems with the interfacing different devices, because these phases will be used during the measurement phase that will be performed through the MATLAB `"port control"` program. The algorithm defines speed, acceleration, and step size for each movement to ensure the movement in the predetermined direction.

3.2.1 AES algorithm

AES has been chosen for this task as the preferred symmetric encryption cryptographic algorithm because it suits this task better, due to its feasibility of upload on this kind of devices. The operating mode will be CBC (Cipher Block Chaining). The algorithm requires to be adjusted in a way that allows the user to acquire as many samples as needed. For this reason, it was necessary to create a modified version of the original CBC mode and rename it AES_SHORT Program 4.

```
// Enable ECB, CTR and CBC mode.
#include <AccelStepper.h>     // run the trigger
int val=0;
#define AES_SHORT 1
#include "aes.h"
#include "aes.c"
#include "test.h"
#include "test.c"
#include <Arduino.h>
```

Program 4. defining AES_SHORT

The adopted algorithm was tiny AES in C [48]. The entire algorithm was uploaded as software in an Arduino Mega 2560 to be used as a portable AES symmetric cipher on an embedded processor. The difficult part is to shape the software in a way that is simple and easily recognizable from the DUT, to avoid the acquisition of different side background operation unnecessary to perform CEMA and DEMA and that can induce noise in the measurements. The algorithm needed is used to run the same operation many times in a loop, allowing the probe to acquire more samples of the same encryption operation.

The cipher can use a defined number of S-boxes and rounds. It is possible to observe from Program 5 that AES 128 uses 4 S-box where each one can store 32-bit output and the number of rounds is 10.

AES_SHORT implementation uses only the first round of S-Box and only *round 1.*

```
// The number of columns comprising a state in AES.
// This is a constant in AES. Value=4
  #elif defined (AES_SHORT) && (AES_SHORT == 1)
      #define Nk 4     // The number of 32-bit words in a key.
      #define Nr 1     // The number of rounds in AES Cipher.
  #else
      #error "must select AES version"
  #endif
```

Program 5. Number of S-boxes

Figure 22Figure 19 shows the entire S-box architecture. The starting part of the loop code involves the area where the S-box table accesses are located. The 32-bit outputs from the S-boxes are *exclusively OR-ed* together to create *t(0)* through *t(3),* then *t(0)* through *t(3)* are shifted and transformed into a 16-state bytes state.

S-Box

```
static const uint8_t sbox[256] = {
  //0     1    2      3     4    5     6     7      8    9     A      B    C     D     E     F
  0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
  0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
  0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
  0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
  0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
  0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
  0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
  0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
  0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
  0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
  0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
  0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
  0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
  0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
  0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
  0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16 };
```

Figure 22. S-Box scheme [48]

AES_SHORT is the modified CBC mode that needs to present a part in the loop code used to make the measurement delayed for being acquired. Through these changes from the original CBC mode, it is possible to acquire the *exclusive-OR-ed* state while before these values were operated through the S-boxes. These bytes are used in the AES operation called ”AddRoundKey”, where the resulting new state is used to address these values into the S-box during the next AES operation. Only the first round is repeated in the loop iteration and at the end of the measurement the AES encryption will proceed in its normal function. Each round of AES encryption is the equal, with the exception that the last round will present the ”AddRoundKey” operation.

Only round *1* of AES encryption is executed in a loop in the Arduino board for a finite number of iterations using the same plaintext as input.

```
#if (defined(CBC) && CBC == 1) || (defined(ECB) && ECB == 1)
static const uint8_t rsbox[256] = {
  0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
  0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
  0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
  0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
  0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
  0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
  0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
  0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,
  0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,
  0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,
  0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
  0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,
  0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,
  0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,
  0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
  0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d };
#endif
```

Figure 23. CBC S-Box scheme [48]

The round constant word array, `Rcon[i]` contains the values to the power *(i-1)* being powers of *x* in the field $GF\ (2^8)$, (Figure 23).

```
static const uint8_t Rcon[11] = {
  0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36 };
```

Figure 24. Most significant byte of the 128-bit input [48]

Only the most significant byte of the 128-bit input is changed in each loop iteration and the other bytes are held fixed (or constant) and they are shown in Figure 24. The value of the most significant byte of plaintext starts at 0 and continues sequentially until it reaches a value of 255 then restarts from 0. Only the most significant byte of the plaintext changes to allow only the first S-box table *Te0* in round 1 to change. All the other rounds from the one S-box are constant. The noise created by this table access needs to be minimized.

The code `AES_SHORT` was modified to repeat the byte utilized in `AddRoundkey` 64 times before going to the next byte. In this way, it should give the oscilloscope the necessary amount of time required to acquire the information as shown in Program 6.

```
// This function adds the round key to state.
static void AddRoundKey(uint8_t round, state_t* state, . . .
. . .const uint8_t* RoundKey) {
  uint8_t i,j,RPM;
  for (i = 0; i < 4; ++i) {
    for (j = 0; j < 4; ++j){
      //REPEAT THE OPERATION 512 TIMES.
      for (RPM=1; RPM<=64; ++RPM){
      delay(2);
      (*state)[i][j] ^= RoundKey [(round * Nb * 4) + . . .
      . . .(i * Nb) + j];
      printf("%.2x", RPM);}
      printf("\n");}
  }
}
```

Program 6. Delayed `AddRoundKey` operation and looped [48].

The delay from the software setting and the trigger were not known exactly but it was possible to estimate them by doing several trials modifying the `RPM` used as a number for the acquisition loop and the `delay(2)` operation. In this way, the sampling time of every operation is fitted into the same sampling window.

3.2.2 Port Control through MATLAB

For controlling the device, the following script has been written to interface MATLAB with the test bench, running several commands and setting up the measurement devices.

```
%% port Control test bench
port = 'COM4';      % test bench communication port
board = 'Uno';
baudrate=9600;
% create a serial port
s=serialport(port , baudrate)
```

Program 7. MATLAB serial port control [32].

Programs 7 and Program 8 show how to interface the different boards.
The launch of the commands into the Arduino boards and how the device responds to an entered command will be demonstrated in Program 8.
It is possible to connect many boards and different ports at the same time.
Program 8 demonstrates the use of the `writeline` command to configure the movement, the acquisition and the trigger [46].

```
%% different write
writeline(s, "I");
n=writeline(s, "2");
%% different read
readline(s)          % read a text
read(s);             % read a number
```

Program 8. Interface commands

The command `readline` as in the previous case is used to read the answer from the device [49].
Program 9 shows the command to initiate the standby phase and return to work, discussed in the previous section.

```
%% Standby
writeline(s, "S");
pause (2);
r=readline(s);
R=(r)
```

Program 9. Standby command.

This command ended up being one of the most necessary and useful of the experiment because it helps to refresh the engines during the dead time, while other parts of the algorithm are working.
The standby command allows the engine to perform the movement and after this turn off the power to the engines, allowing the motor to cool down during the scanning and to avoid overheating of the scanning device.

3.2.2.1 Test 1

The first test model is used to scan an area of interest. When launched, a request appears for entering a height value and a length value to start scanning a user-defined area. The variables are differentiated with the respective number for both tests, so that

they can be launched simultaneously, reducing the scan time, and limiting it only to the sections of interest. This scan was very useful in drawing the heatmap and studying the leakage point of the device.

```
% Define one area in length and height
clc;
l1=input('Value of length: ');
c1=input('Value of height: ');
d1=input('Value of samples: ');
Y1=0; % counter for height Y
i1=0; % counter for Length X
k1=0; % return counter for Height x
y1=0; % return counter for Height
C1=1;
L1=1;
```

Program 10. Initialization Test parameters

Program 11 shows the scanning path in length and height of the bench, in this way, the entire microchip area can be reached. With the command `picocapture` [21] the oscilloscope acquires the information from that point.

```
%% Scan Area (Test 1)
for Y1=1:1:c1 %% position 0 Y1=1:1:c1
   writeline(s, "s");
   pause(2);
   R=("count column ")
   R=(Y1)
   C1=(c1);

```

Program 11. Initialization Test 1

Program 12 shows how the variables can be input into the MATLAB console from the user.

```
for X1=1:1:l1 %% position 0 X1=1:1:l1
    writeline(s, "d");
    pause(2);
    writeline(s, "S");  %
    pico_capture(X1, Y1, d1);
    R=(X1);
    L1=(l1);
     if L1==X1
        writeline(s, "U");  % turn on
     for x1=1:X1
         writeline(s, "a");
         pause(1);
     end
     pause (2); %% wait 2 sec before acquiring again.
     L1=0;
     end
```

Program 12. Scanning algorithm length and height.

In the loop, the algorithm will autonomously direct the system scan along the length X and wait for 2 seconds at each movement, continuing along its path in a straight line. When it reaches the end of the run it will reverse at a higher speed without pauses (1 second to keep track of the steps) to continue one step in height Y.

These steps proceed until the entire scanning area is covered, and the last part of the code in Program 13 will make the bench return to its original acquiring position.

```
if C1==Y1
for y1=1:Y1
% Need to come back 1 step more
   writeline(s, "w");
   pause(1)
end
C1=0;
Writeline (s, "S");  %turn off
```

Program 13. Returning to Origin.

Test 1 is a good way to perform a sweep scan of an area of interest or across the entire surface of the device because all lengths are rounded in *mm* by step size. This is a useful procedure to start with an exploratory approach to the device and to define the first MTD.

3.2.2.2 Test 2

The second test is carried out to move the device directly to a microcell of interest, it is possible to define the value in length and height of the position of interest just as in `Test 1`. But it is different in the sense that it is possible at first to define a point of scanning after launching `Test 1` to acquire a micro area of interest. `Test 2` was implemented to work with the AES code [48] and to acquire the CEMA and DEMA EM SCA traces.

```
%% Scan Point (Test 2)
clc;
l2=input('Value of length: ');
c2=input('Value of height: ');
Y2=0; %counter for height Y=j
X2=0; %counter for Length X=i
x2=0; %return counter for Height x=k
y2=0; %return counter for Height
C2=1;
L2=1;
```

Program 13. Initialization Test parameters

It works quite similar to *`Test 1`*, but when it reaches the point, it goes into standby mode, in this way it is possible to operate something else in the meanwhile.

```
%% Scan Point (Test 2)
for Y2=1:c2
   writeline(s, "s");
   pause(2);
   R=("count column ")
   R=(Y2)
   C2=(c2);
end
for X2=1:l2
   writeline(s, "d");
   pause(2);
   R=("count line ")
   R=(X2)
   L2=(l2);
end
```

Program 14. ***Initialization Test 2***

The operation of trigger enables the acquisition of the AES code, which begins with the input of the number of traces that the user needs to gather.

```
   %% Acquisition + Trigger Operation
% Before launch a read trigger to read the ready command
clc;
X1=20;
Y1=16;
A=0;
i=1;
d1=input =("value of sample");
writeline(so, "t");
   pause(2); % Begin the test
r=readline(so);
R=(r);
pico_capture (X1, Y1, d1, so);
```

Program 15. ***Acquisition and Trigger***

The trigger begins with a `writeline` command, and this operation coordinates the beginning of the encryption and the beginning of the acquisition. The operation of trigger is reported in the MATLAB console through the command *readline*.

```
%% Trigger operation
clc;
writeline (so, "t");

%% Read Trigger operation
r=readline(so);
R=(r)
```

Program 16. ***Trigger command***

The trigger command begins when the user inputs a `writeline` command *"t"* to run the operation command in Arduino IDE. The `read trigger operation` will report as first notification stating that the trigger command is `ready` and that can begin through the command. The second command will begin the EM SCA analysis through MATLAB. When it is no longer necessary to remain in the position of interest, the bench will return to the initial position so that a new position command can be issued, the system does not need to be recalibrated.

```
%% Return to initial position from the Point (Test 2 return)
for x2=1:X2
    writeline(s, "a");
    pause(1);
    R=("return fast length ")
end
L2=0;
for y2=1:Y2
   % Need to come back 1 step more
    writeline(s, "w");
    pause(1);
    R=("return fast height ")
end
C2=0;
```

Program 17. ***Test 2 return Command.***

Using the script in Program 17 at each end position it is possible to avoid the repositioning of the bench at the origin of the scanning.

Test 1 automatically re-runs a phase of return to the initial position.

```
%% end of transmission
clear so;
clear s;
("Port Dismiss") %to disconnect and clear the serial port
```

Program 18. ***Disconnect the serial port.***

Program 18 shows how to close the connection with the serial port [46].

3.2.2.3 Script to launch `pico_capture`

`Picocapture` is the name of the function used to launch the software released by *`PicoScope 5000`* [21] series in MATLAB which allows the oscilloscope to be used as a device directly from MATLAB script.

```
function pico_capture(X1,Y1,d1, so)
for Z1=1:d1
r=readline(so);
fprint(r)
```

Program 19. Pico_capture function

Program 19 shows the function used to acquire the electromagnetic traces from the operating board. The function is written to work as an area test bench; *(X, Y)* is the position of acquisition and the *Z* value is the number of samples acquired. Function *Z* does not get confused with the turret Z of the bench where the probe is attached. The parameter `d1` and `s0` were necessary to launch the function during the trigger operation.

```
%% LOAD CONFIGURATION INFORMATION

PS5000aConfig;
folder_name = 'path_to_folder'
file_name = string(X1) +'_'+string(Y1) +'_'+string(Z1) +'.mat'
```

Program 20. ***Information to configure the data.***

The parameter `s0` is the name of the function to use the operational board and so launch the encryption phase with AES; *d1* is the parameter used in the function to define the number of samples to acquire; that number, that in this case, is the Z coordinate.
The oscilloscope needs to be configured not only as a device but also it needs to know where the information needs to be stored. The oscilloscope gathers the samples, through

the values *X, Y, Z,* and these values, which represent the point of acquisition, make it possible even to discern one acquisition from another.

In Program 22, the folder name is *"measure",* (It is worth noticing that before calling the folder it is necessary to specify the path of the folder and how the information is saved). Every *pico_capture* trace was called with the name of the segment, from where it was collected.

When the program launches in the console, it asks the user to specify the number of samples for d1. Later, the program will acquire the traces from the two different settings: *pico_capture* for *Test1* or *pico_capture* for *Test2*. Both are very similar but differ in the kind of acquisition they make, moreover:

- *pico_capture* for *Test1*: It is used to acquire EM traces in order to draw a heatmap as shown in picture 10, executing a total excursion of the microchip.
- *pico_capture* for *Test2*: It is used to acquire the EM traces necessary to perform CEMA and DEMA as shown in picture 6,7,8 and 9.

Every trace in both cases specifies the *X, Y, Z,* as name. Example: *1_1_1* means trace acquired at the first point X, first point Y, first sample of the collection, and so on until reaching the total excursion of the chip for *Test 2* or the whole collection of the leakage information from the point chosen as $Area_a$, where the maximum number of leakages are emitted.

It is necessary to set this up just to have a break from one trace and the next.

Pre-trigger was settled to 1024 samples which is the maximum number of samples to

```
% Set pre-trigger samples.
set(ps5000aDeviceObj, 'numPreTriggerSamples', 1024);

[status] = invoke(blockGroupObj, 'runBlock', 2);
```

Program 21. ***Pre-trigger***

define the break from a block of traces and the next, enabled by the oscilloscope.

To retrieve the data for further analysis like heatmaps and SCA analysis, it was necessary to add the command *save*, to save file with a name and move the block trace in a label of destination with the command *movefile* in the path of destination.

```
movefile (string(X1)+'_'+string(Y1)+'_'+string(Z1)+'.mat',
'C:\Measure'+string(X1)+'_'+string(Y1)+'_'+string(Z1)+'.mat' )
```

Program 22. ***Pre-trigger and label of gathering.***

The last command is necessary to disconnect the devices and avoid MATLAB issues.

```
%% DEVICE DISCONNECTION
%(write this in the console to restart the Device)
instrreset;
```

Program 23. ***Devices disconnect.***

Device disconnection was necessary to avoid error of initialization of the serial port and to help in some oscilloscope issues. Before beginning a real acquisition, it is necessary to find the right area and to set it up properly. With this command it is possible to avoid the restart of MATLAB and resume the measure in a few seconds.

3.2.3 Operations for result post-processing

This part of the paragraph is dedicated to the MATLAB script written as a complementary part of the proposed driver. These algorithms for processing results can be used to analyze the acquired traces in a few fast operations.

3.2.3.1 Test 1 and heatmap plot

The script in Program 24 presents the results gathered with the trace acquisition of `Test1.` This algorithm draws the heatmap of the cryptographic device under investigation. *X1* and *Y1* are necessary to begin the sampling operation and process all the traces in the right order of acquisition until reaching the full processing of all samples. Every point of the chip is gathered in the array *fulldata=[].* In the array *fulldata* the entire line is recorded in different samples that go from 1 to 20, in a total number of samples per set record (Z). All the samples gathered in the array will be processed through a full chart of the chip, *meanvalueSNR* [50] operation is necessary to process all these traces to recognize where high leakage emission points are located from the microchip.

```
dir='C:\Users\Measure_1_32x31x20 op1\';
for X1=1:32
    for Y1=1:31
        fulldata=[]
        for Z1=1:20
        path=dir+string(X1)+'_'+string(Y1)+'_'+string(Z1)+'.mat'
        data=load(path);
        data=data.chA;
        fulldata=[fulldata; data];
        end
        meanvalueSNR= snr(fulldata);
        ArrayOfVal(X1, Y1)=meanvalueSNR;
```

Program 24. *Heatmap*

The command *h=heatmap(ArrayOfVal)* [51] plots the heatmap in Figure 25 with the exact order of acquisition of the samples and shows the values of every segment of the chip Program 24.

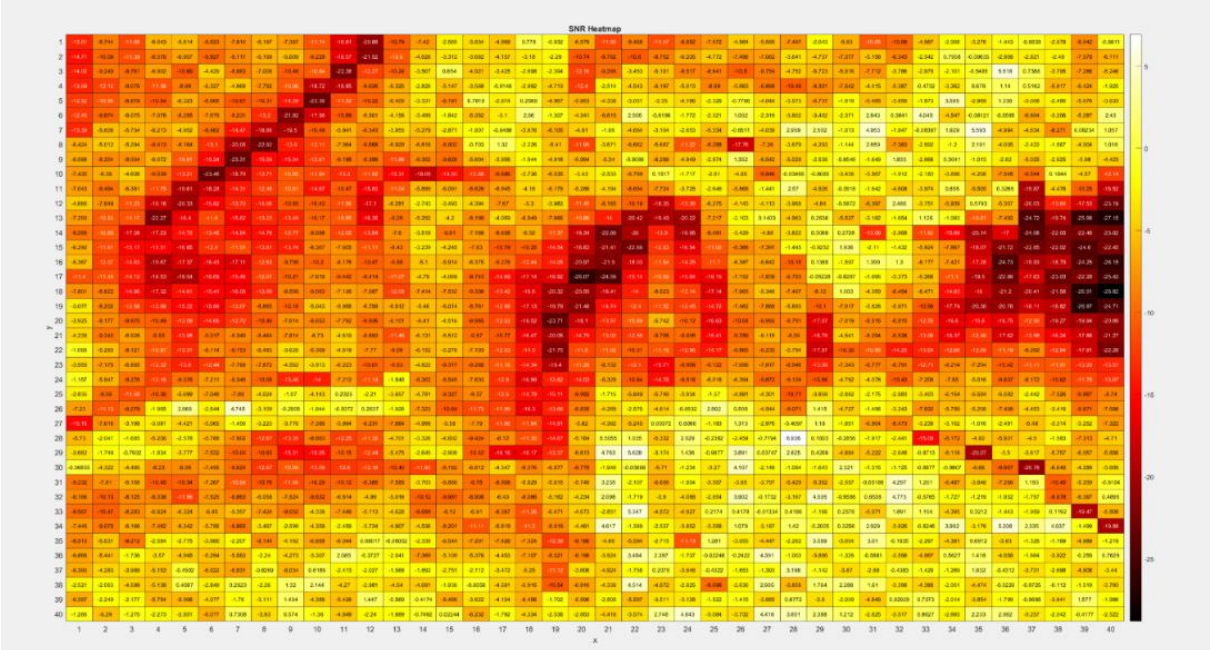


Figure 25. Heatmap plot

The script for differential analysis was based on the demonstration shown in the chapter literature, on the acquired traces from Test 2 and the AES algorithm implemented into the operational board.

3.2.3.2 Differential analysis of test 2 results

The values *s0* and *s1* are numeric arrays into character arrays that represent the number, with a special format (*formatSpec*), in this case, *formatSpec="%.0f"* is a fixed notation for a flag that specifies the number of digits after the decimal point. The value *s0* is the current element of the string (or the first number if it is one) and s1 is the next in the next in the following number of the string. *μ0* is the mean value of *s0* and *μ1* is the mean value of *s1* (Program 25).

```
function x = Differential_Analysis()
for j = 1 : 131
    formatSpec = '%.0f';
    s0 = num2str(j, formatSpec);
    s1 = num2str(j + 1, formatSpec);
    name = strcat('Measure_20x16_132_Addroundkey\20_16_', s0,
'.mat');
    disp(['Loading ', name])
    elements0 = load(name);
    name = strcat('Measure_20x16_132_Addroundkey\20_16_', s1,
'.mat');
    disp(['Loading ', name])
    elements1 = load(name);
    u0 = mean(elements0.chA); % μ_0^{j,K}(t)
    u1 = mean(elements1.chA); % μ_1^{j,K}(t).
```

Program 25. ***Initialization of values from the traces***

Program 26 shows the implementation of the equation mentioned in chapter 2.1. In literature is called the differential trace $D^{j,K}(t) = \mu_0^{j,K}(t) - \mu_1^{j,K}(t)$ (2.1). It is used to compile the $s = \sqrt{\left(\frac{1}{n_0}\right)\left(\sigma_0^{j,K}(t)\right)^2 + \left(\frac{1}{n_1}\right)\left(\sigma_1^{j,K}(t)\right)^2}$ (2.2) that is called a *significant differential trace.*

```
% Differential trace
Djk = u0 - u1;
n0 = length(elements0.chA);
n1 = length(elements1.chA);
Sigma0 = 0;
% σ_0^{j,K}(t)
Sigma1 = 0;
% σ_1^{j,K}(t)
for i = 1:n0
    A = elements0.chA(i) - u0;
    Sigma0 = (Sigma0 + (1/n0)*A^2);
end
% σ_0^{j,K}(t) part 3 Final result.
Sigma0 = Sigma0^(1/2);
for i = 1:n1
    A = elements1.chA(i) - u1;
    Sigma1 = (Sigma1 + (1/n0)*A^2);
end
```

Program 26. ***Differential trace***

Program 26 shows the initializing differential trace and significant differential trace.

The value of $\sigma_0^{j,K}$ that is `sigma 0` and of $\sigma_1^{j,K}$ that is *`sigma 1`* are root squared as shown in Program 27.

Equation 2.2 is implemented in Program 26.

$$D_S^{j,K}(t) = \begin{cases} D^{j,K}(t) - 2s, if\ D^{j,K}(t) > 0 \\ D^{j,K}(t) + 2s, if\ D^{j,K}(t) < 0 \end{cases}$$

Program 27 shows the differential trace 2.1 implemented with both limits if it is positive and negative as presented in the demonstration.

```
Sigma1 = Sigma1^(1/2);
    %standard deviation of the differential means (s)
    s =( (1/n0)*(Sigma0^2) + (1/n1)*(Sigma1^2))^(1/2);
    % significant differential Trace
    if Djk > 0
        disp(['Djk > 0 (', num2str(Djk), ')'])
        Djks = Djk - 2*s;
    else
        disp(['Djk < 0 (', num2str(Djk), ')'])
        Djks = Djk + 2*s;
    end
```

Program 27. *Significant differential trace*

The last part of this script is the setup to configure the plot. The idea is to implement a plot that shows the entire area of the acquisition with all the traces and a more magnified presentation of the peak of interest, which represents the actual value written in the S-box.

3.2.3.3 Correlation analysis of test 2 results

Correlation analysis uses the formulas for the Hamming weight W_i^K (2.4), in this case: $W_i^K = \frac{E_{C,W}^K(t) - \mu_C\,(t)\;\mu_W^K}{(\sigma_C(t)\;\sigma_W^K)}$.

W_i^K is a useful operation that is calculated from $H_K = max_t \left|D_s^{j,k}\,(t)\right|$ (2.3), this value is the *height of the differential* obtained as the result of performing `Differential Analysis` script (Program 28) and it is obtained as a key guess K, that is a decimal number. W_i^K that is the result of *Correlation analysis as Correlation factor at time t based on the key guess*, will be obtained as the last result. In this script, the H_K value is used as a variable value in the function `Correlation_Analysis` and launched from the differential frequency analysis script. The first operator involved with the Hamming weight is the $E_{C,W}^K(t)$ that is the expected value of the of $C_i\,(t)$ that are the samples and W_i^K.

This operation is performed in the first cycle (`for`), with $\mu_C\,(t)$ and μ_W^K from (2.5) in Program 28.

```
function Correlation_Analysis(HMax)
n = 132;
rhoK = zeros(n, 1); % ρ^K(t)
for j = 1 : n
    formatSpec = '%.0f';
    s1 = num2str(j, formatSpec);
    name = strcat('Measure_20x16_132_Addroundkey\20_16_', s1,
'.mat');
    samples = load(name);
    samples = samples.chA;
    WKi = sum(dec2bin(samples + HMax) == '1',2).';
    EKCW_t = 0;
    uC_t = 0;
    uKW = 0;
    for i = 1 : length(samples)
        Ci_t = samples(i);
        EKCW_t = EKCW_t + (1/n)*(Ci_t*WKi(i));
        uC_t = EKCW_t + (1/n)*(Ci_t);
        uKW = EKCW_t + (1/n)*(WKi(i));
    end
```

Program 28. Correlation analysis explanation of the first values

The second *for* cycle is necessary to calculate the two variances as shown in Program 29.

Variance of the value $C_i(t)$ and $\mu_C\ (t)$: $\sigma_C(t) = \sqrt{\frac{1}{n}\sum_i\left(C_i(t) - \mu_C\ (t)\right)^2}$

Variance of the value W_i^K and μ_W^K: $\sigma_W^K = \sqrt{\frac{1}{n}\sum_i\left(W_i^K - \mu_W^K\right)^2}$

```
% function Correlation Analysis σ_C(t)
sigmaC_t = 0;
    sigmaKW = 0;
    for i = 1 : length(samples)
        Ci_t = samples(i);
        sigmaC_t = sigmaC_t + ((Ci_t-uC_t)^(2));
        sigmaKW = sigmaKW + ((WKi(i)-uKW)^(2));
    end
    sigmaC_t = (1/n)*sigmaC_t;
    sigmaKW = (1/n)*sigmaKW;
    sigmaC_t = sigmaC_t^(1/2);
    sigmaKW = sigmaKW^(1/2);
    rhoK(j) =(EKCW_t - uC_t*uKW)/(sigmaC_t*sigmaKW);
end
```

Program 29. Correlation analysis explanation of the variance's values and correlations

Once that the values from the two variances are obtained, it is possible to calculate the $\rho^K(t)$ that is *the correlation factor at time* t *with the key guess K.*

The value $H_K = \max_t \{|\rho^K(t)|\}$ to obtain the *height of the differential* requires the $\rho^K(t)$ found in the previous equation (2.4) $W_i^K = \frac{E_{C,W}^K(t) - \mu_C(t)\,\mu_W^K}{(\sigma_C(t)\;\sigma_W^K)}$.

The value $|\rho^K(t)|$ is retrieved in line 13, at the very last part of Program 29.

3.2.3.4 Frequency-based analysis of test 2 results

Differential frequency-based analysis (DFA) uses the value $H_K = max_t \left|D_s^{j,k}(t)\right|$ (2.3). This value is the result of performing EM DEMA *(t)* as mentioned in correlation analysis. Frequency-based DEMA differs from DEMA in the context of transforming the $D_s^{j,k}(t)$ in $D_j^K(f)$ with the function Fast Fourier Transform (FFT) [52].

```
function HKj = Differential_Frequency_Analysis(HMax)
n=131;

DKj_fMax = zeros(n, 1);
DKj_f = zeros(n, 1);
HKj = zeros(n,1);
for j = 1 : n
    formatSpec = '%.0f';
    s1 = num2str(j, formatSpec);
    name = strcat('Measure_20x16_132_Addroundkey\20_16_', s1,
'.mat');
    disp(['Loading ', name])
    samples1 = load(name);
    samples1 = samples1.chA;
    s2 = num2str(j+1, formatSpec);
    name = strcat('Measure_20x16_132_Addroundkey\20_16_', s2,
'.mat');
    disp(['Loading ', name])
    samples2 = load(name);
    samples2 = samples2.chA;
```

Program 30. Differential frequency analysis

The values of H_K founded in Program 30 are loaded as in the differential analysis script. In Program 31 the $C_i^b(t)$ is the EM signal of plaintext and key guess $\{S(p_i \oplus K)\}_j$. This operation is executed through the parameters *CB_t0* and *CB_t1*.

In Program 31 *CB_t0* and *CB_t1* are initialized as a collection of records of the samples of the mean value increased using the value *HMax* and normalized with the absolute value (abs) [53] of Pb0_fMax and Pb1_fMax in order to obtain $\left|D_S^{j,k}(t)\right|$.

Program 32, computes $H_K = max_t \left|D_S^{j,k}(t)\right|$, which is a necessary step to calculate $P_i^b(f) = \frac{\left|FFT\left(C_i^b(t)\right)\right|^2}{n}$, that is the FFT of all samples of the plaintext in Program 31.

```
% plaintext and K are changed by signals

    CB_t0 = samples1 + HMax;
    CB_t1 = samples2 + HMax;
    CB_t0Max = max(samples1) + HMax;
    CB_t1Max = max(samples2) + HMax;
    Pb0_fMax = (abs(fft(CB_t0Max)).^2)/n;
    Pb1_fMax = (abs(fft(CB_t1Max)).^2)/n;
    Pb0_f = (abs(fft(CB_t0)).^2)/n;
    Pb1_f = (abs(fft(CB_t1)).^2)/n;
```

Program 31. Fast Fourier Transform

```
 sumPb0_f = 0;
for indexPb0 = 1:length(Pb0_f)
    sumPb0_f = sumPb0_f + Pb0_f(indexPb0);
end
sumPb1_f = 0;
sumPb1_fMax = 0;
for indexPb1 = 1:length(Pb1_f)
    sumPb1_f = sumPb1_f + Pb1_f(indexPb1);
end
sumPb0_fMax = 0;
for indexPb0 = 1:length(Pb0_fMax)
    sumPb0_fMax = sumPb0_fMax + Pb0_fMax(indexPb0);
end
sumPb0_fMax = 0;
for indexPb0 = 1:length(Pb1_fMax)
    sumPb0_fMax = Pb1_fMax + Pb1_fMax(indexPb0);
end
DKj_f(j) = (sumPb0_f - sumPb1_f)/(n/2);
DKj_fMax(j) = (sumPb0_fMax - sumPb1_fMax)/(n/2);
HKj(j) = sum(DKj_f);
end
```

Program 32. Differential traces in FFT

The differential traces under the Fourier domain are calculated with equation 2.6.

This equation, $D_j^K(f) = \frac{1}{\frac{n}{2}}\sum_i \ P_i^0(f) - \frac{1}{\frac{n}{2}}\sum_i \ P_i^1(f)$. is processed in Program 32.

The last equation is the H_j^K *sum(Σ) of all differences in amplitudes over all the frequencies* (2.7). $H_j^K = \sum_f D_j^K(f)$, compiled in Program 32.

4. RESULTS

This chapter will discuss the results obtained in this thesis. It will present how the heatmaps are generated, what they show, why they are necessary, and why they are used. The next part will be focused on the differential analysis, how the results were gathered and calculated with the software and how they can be understood. The same procedure will be used for the correlation analysis and the differential frequency analysis. Figure 26 presents the architecture of the DUA (the device under attack), that is, the Arduino board microchip 2560. This microchip represents the personal computer device (PCD) and even the machine that is used to perform portable encryption as symmetric cipher for the AES cryptographic algorithm.

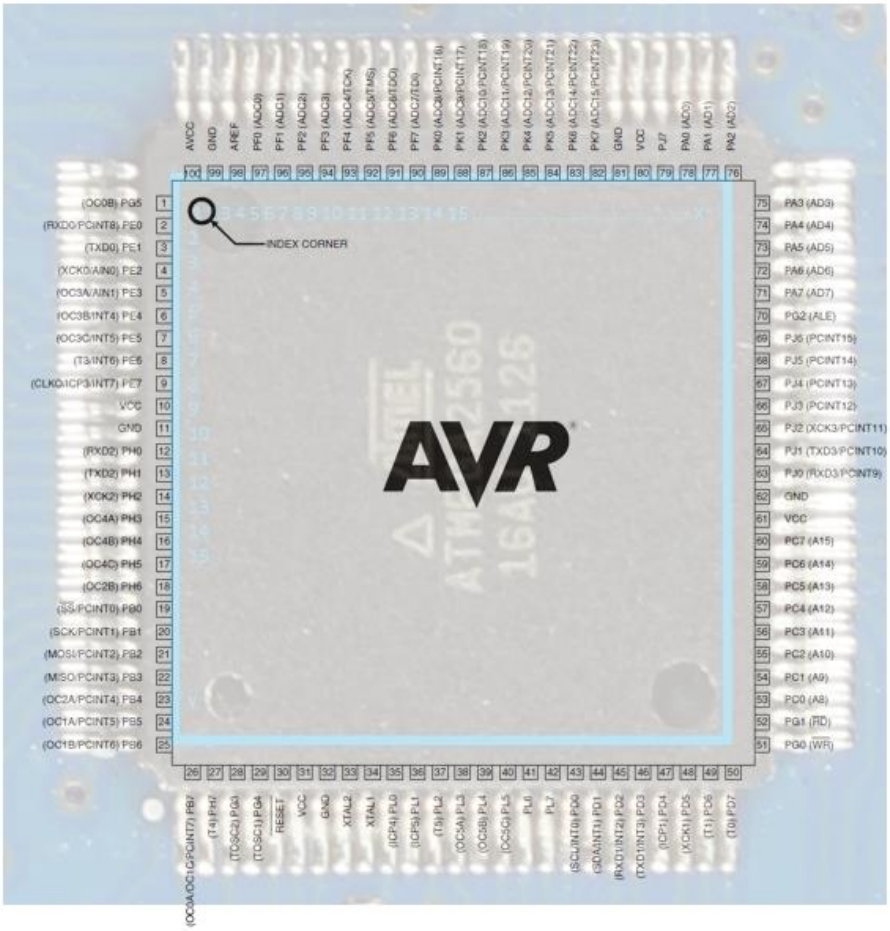


Figure 26. Arduino board microchip 2560

In Figure 26 the acquired segment of the microchip with the measurement bench is highlighted in a light-blue rectangle. To better understand the heatmaps, the microchip schematic is depicted in Figure 27. It is another useful diagram which describes how the pins

are connected between themselves and can help to characterize where the leakage is located. The region of leakage will be discovered and chosen from the attacker, becoming the targeted RoAs area.

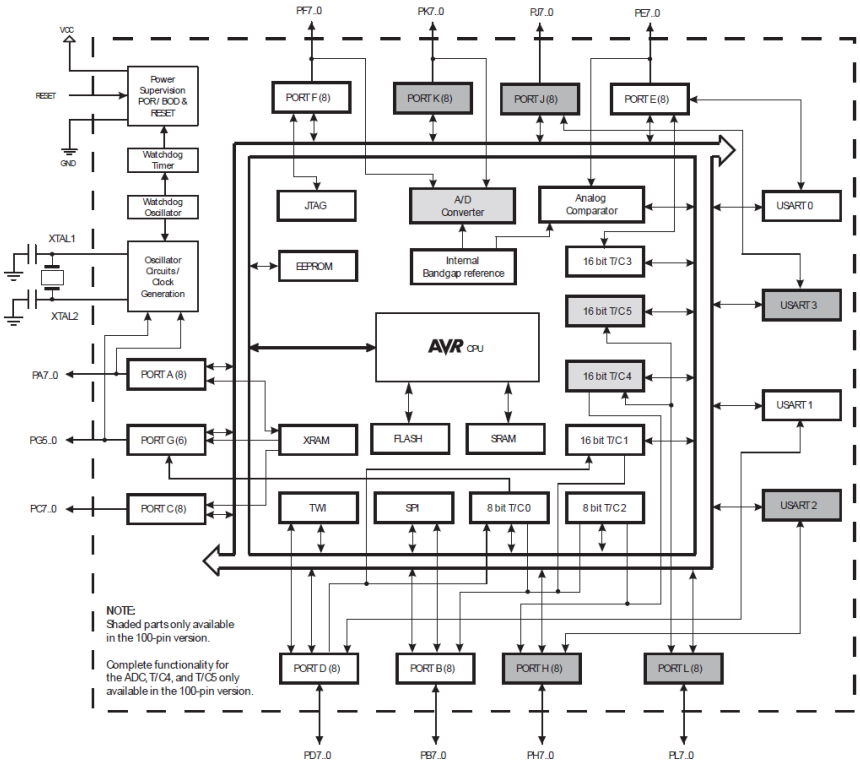


Figure 27. Flux diagram of the microchip 2560 chip.

4.1 Heatmap

As described in *section 2.6,* it is necessary to have a procedure to characterize a device. By Launching `Test 1`, it is possible to acquire all the EM traces of the whole program and the heatmap shows exactly which areas have maximum leakage point. As described in point three of the characterization procedure, it is necessary to understand which is the Region of attack (*RoA*). It is possible to affirm from Figure 29 that there are three possible RoAs on the microchip, the black spots that are the points at maximum leakage are gathered more in those three regions:

1. in the upper right, *region 1*
2. in central left, *region 2*
3. in the central bottom, *region 3*

With Figure 28, it is possible to link the points of maximum leakage with the hot spot regions of the microchip from Figure 29:

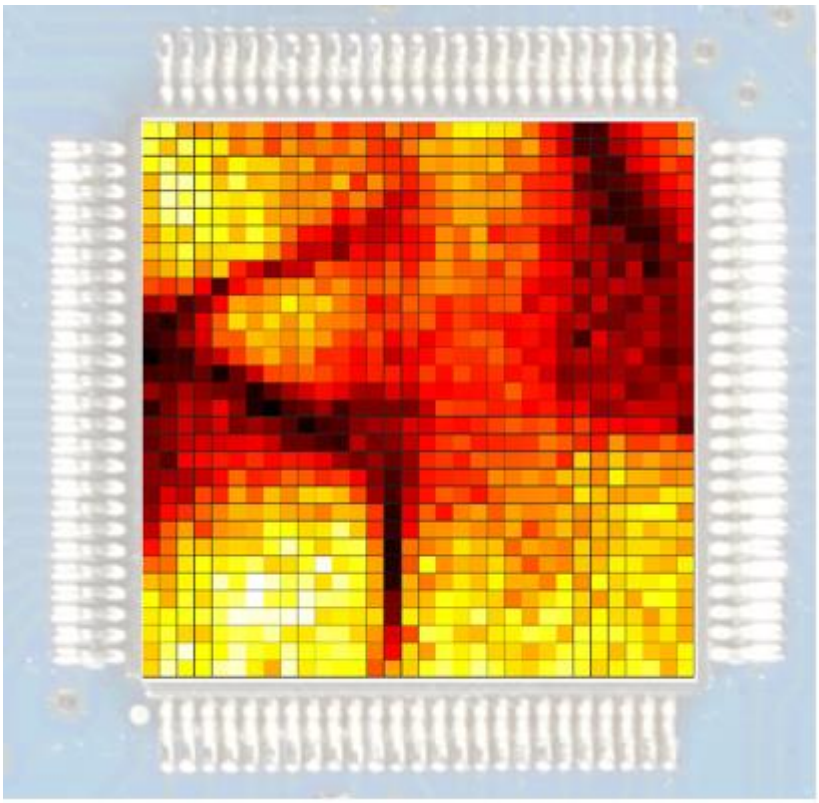


Figure 28. Heatmap from the probe less accurate

Region 1: A/D converter, Analog Comparer, internal bandgap, 16-bit T/C 3 and 16-bit T/C 5.

Region 2: EEPROM and RAM.

Region 3: Flash memory.

With this result it was possible to assume that the device was working as expected, because it gathers the information from the ALU, where the A/D is positioned and so where the operations were performed, from the EEPROM where the information are stored, and from the flash Memory that is the line in the inner part of the microchip that is the place where the information were sent to the communication bus.

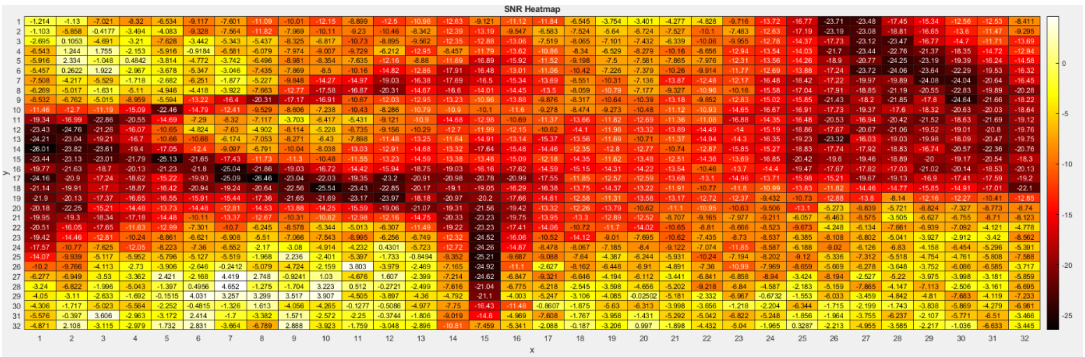


Figure 29. Heatmap with value in dB of the SNR

The less accurate probe has given, as expected, a heatmap with more leakage points, so providing major uncertainty of the hot spot areas. The aim of the accurate probe used later is to provide a more precise measurement, reducing the RoA where leakage points are accurately identified. Figure 30 presents the heatmap using the more accurate probe. It is possible to observe that the leakage points are more focused in the actual maximum leakage points and the RoA are narrowed.

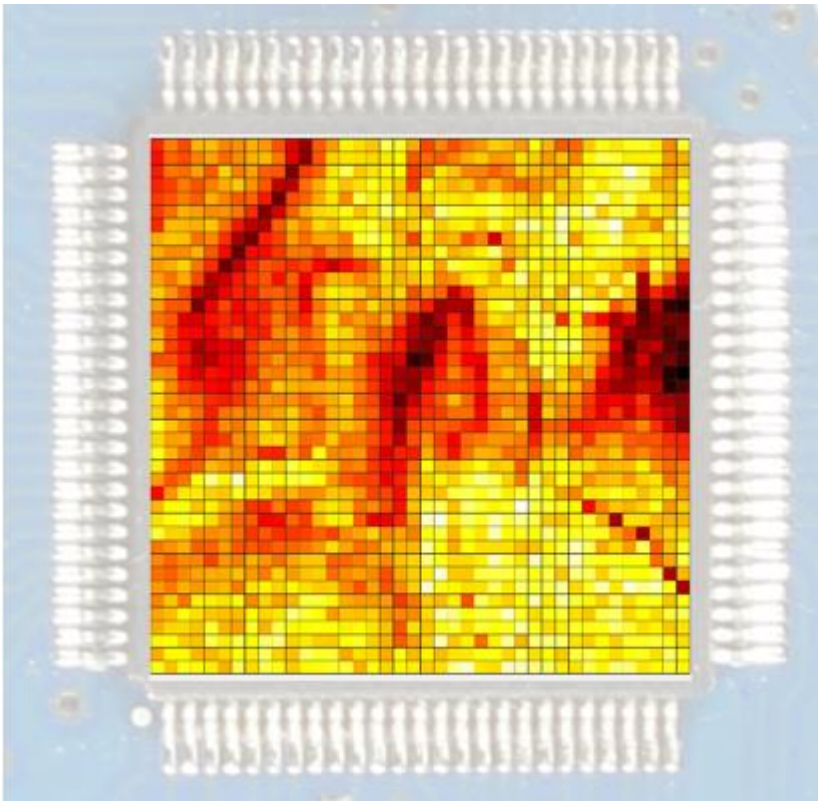


Figure 30. Heatmap from the accurate probe

In Figure 30 the RoAs are more focused with fewer black spots.

The reasons for this are:

Region 1: Does not highlight all the different areas of the devices but shows that more leakage points are positioned in the areas of the *16-bit T/C 3 and 16-bit T/C 5.*

Those points present: *T/C 3 of -27.15 dB* and *T/C 5 of -28.82 dB.*

Region 2: EEPROM and RAM are precisely highlighted, the black dots are very few, but if it was chosen as preferred RoA, it can be considered a good choice where to launch `test 2`, used for a fixed point with the test bench. Point (5,10) presents an SNR of -23.36 dB, and it could be defined as a point at high leakage for the memory area.

Region 3: Flash memory was my point of interest because it was highlighted from both probes. I decided to launch `Test 2` exactly in the central area of the microchip; at point (17,20). This point is not only the place that accumulates more leakage activity, but it is

also the place that connects the communication bus, and the place where the `AddRoundKey` bits travel to be stored in the S-BOX memory.

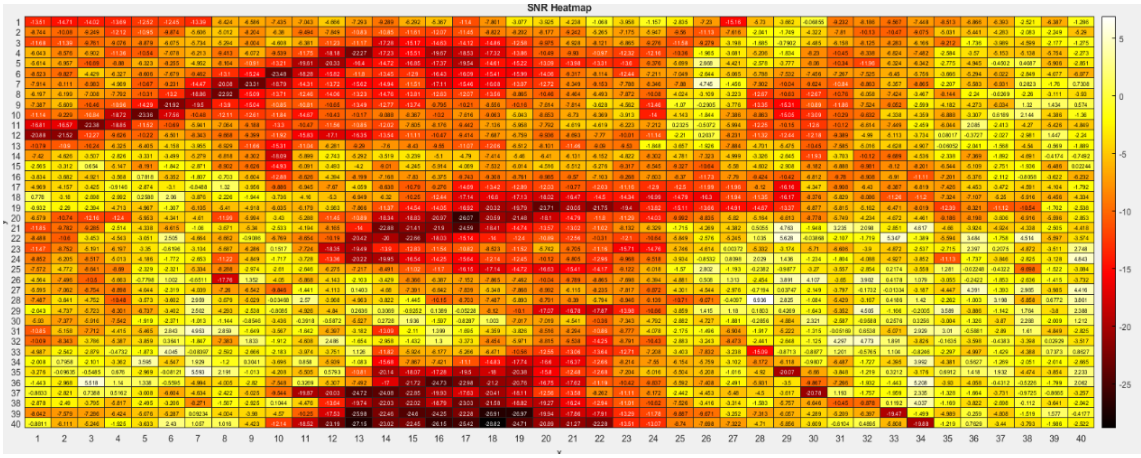


Figure 31. Heatmap from the accurate probe

The point that intercepts the *Flash memory* in position (17, 20) presents a leakage value of -26.07 dB. The Figure 30 is rotated in comparison with Figure 30 because the measuring bench had a problem where the accurate probe was positioned on few pins of the operational board, and it was necessary to remove the pins from many areas of the boards, to allows the acquisition of the measurement.

4.2 Differential analysis

Differential analysis was performed during the `AddRoundKey` operation at the output of the S-Box, that in the case in the heatmap is the position (17, 20). The point is the flash memory location and corresponding to the area where the information travels from the memory to the communication pins. This procedure requires that the attacker will know the plaintext values and the S-box table values. For the proposed case solution, it should be necessary to do a separate study of how the plaintext and the key are *exclusively-OR-ed* together and understand every reported position of bit flip from 0 to 1 and vice versa, to discover the key.

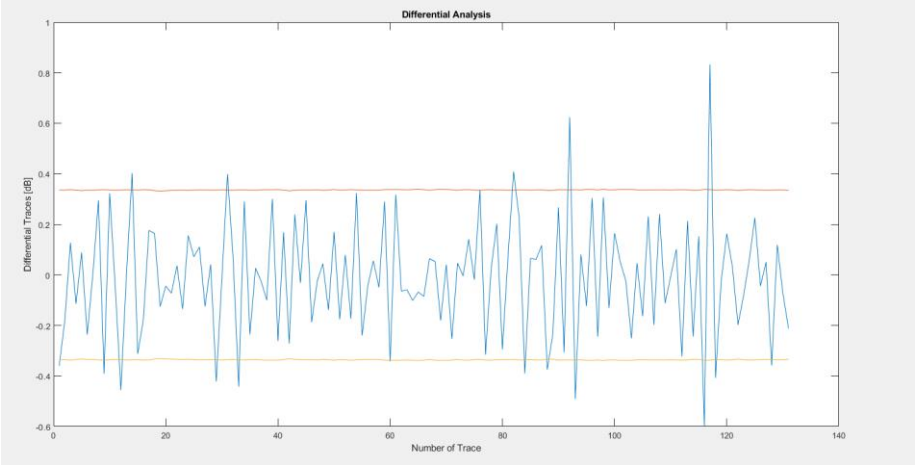


Figure 32. Differential analysis plot

The trace acquired from the probe is shown in Figure 32. The top red line and the bottom yellow line describe the significant differential traces. These lines define the limit threshold, where the values that exceed the lines are the differential traces that can be divided in two ways. The differential trace needs to present two spikes:

1. A low spike followed by a high spike representing a plaintext load, which means, a flip from a zero-bit to a one-bit.
2. A high peak followed by a low peak is the S-box load, and it is represented as flip from a one-bit to a zero-bit.

The higher differential traces are the key operations traces. In Figure 32 it is possible to recognize many operations of plaintext load and S-box load. They can be identified in groups because the two loads are following one each other after a few oscillations.
The next part of this paragraph will be more focused on how the differential traces need to be read and how to discern between those in order to define the operation of load for plain text and for the S-Box. From the results it was possible to divide the operations of loading into two group of traces that are shown in two Figure 33 and Figure 34.

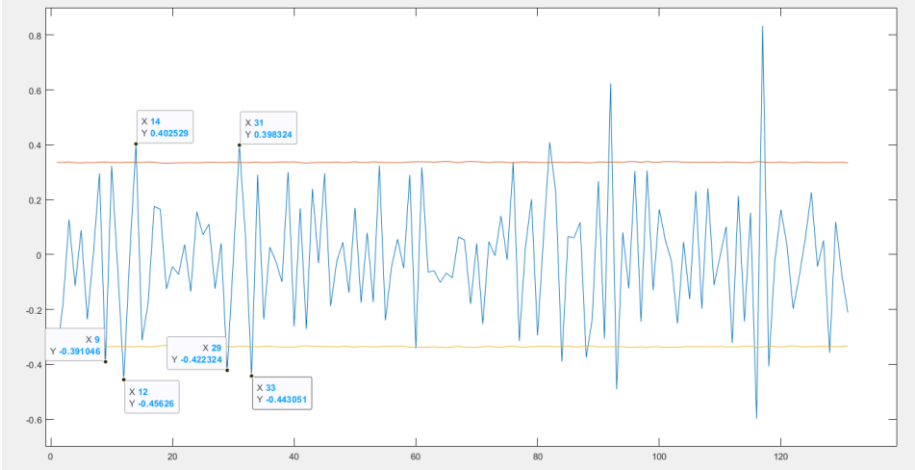


Figure 33. First load group

In Figure 33 the plaintext loading can be seen in the group traces that have axis value of 9, 12, and 14 the more magnified value is the point (12, -0.456) which is located at position 12 of the graph and present a magnitude of -0.456 dB. Repeating the same procedure, it is possible to discern from the group of traces 29, 31, and 33 as a load to the S-Box, where the more magnified point is at (31, 0.393) which means, that traces 31 present a magnitude of 0.398 dB, and this is an S-Box load.

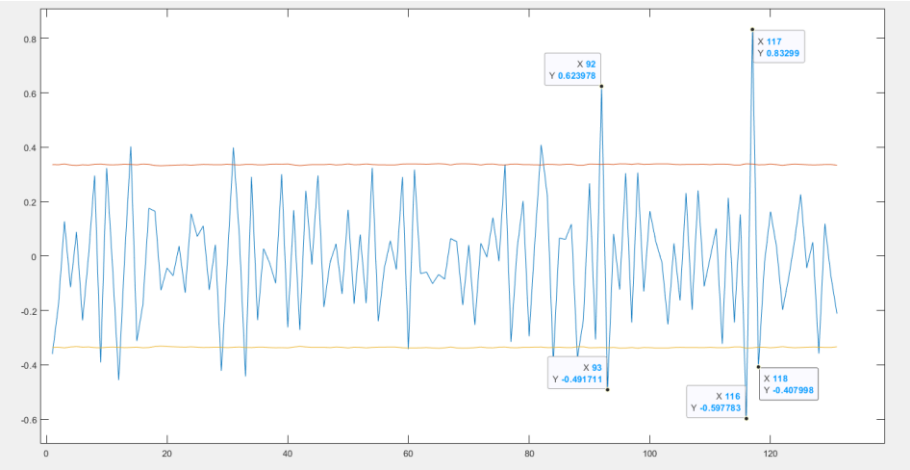


Figure 34. Second load group

From Figure 34 it is possible to recognize a similar situation that is recorded in traces 92 and 93 where the plaintext load is located, where the more magnified point is (93, -0.491), that means located in axis 93 and present and magnitude of 0.456 in dB.

The S-box load is gathered at traces 116, 117 and 118. The most magnified point is at (117, 0.832) where the point is located on the graph at axis 117 and presents a magnitude of 0.832 dB.

# Number of the trace	Magnitude (dB)	Plaintext Load	S-box Load
9	-0.391		*x*
12	-0.456	*X*	
14	0.402		*x*
29	-0.422	*x*	
31	0.398		*X*
33	-0.543	*x*	

Table 2. First load group values

Table 2 gathers all points from Figure 33 and Figure 34, It is possible to observe that all the points show a clear description of the Plaintext load *(X – higher amplitude in comparison with x – smaller amplitude)* for the trace 12 and as second point trace 31 close

the cycle loading *(X – higher amplitude, x – smaller amplitude)* into the S-box. From Figure 32 the peaks are very clear and recognizable.

# Number of the trace	*Magnitude (dB)*	*Plaintext Load*	*S-box Load*
92	*0.623*		*x*
93	*-0.491*	*X*	
116	*-0.597*	*X*	
117	*0.832*		*X*
118	*-0.407*	*x*	

Table 3. Second load group values

In comparison with Table 2, Table 3 presents values higher in intensity. The peaks in Figure 34 are more intense than Figure 33. Unfortunately, the noise and maybe some hidden operations could have changed the expected results. But from this preliminary presentation the system has given an expected behaviour standing to the already known theory proposed in chapter 2.

4.3 Correlational analysis

From the correlation analysis executed at the output of the S-Box, the key guessing values are reported in the five points of interest, from the differential analysis first and second load groups. In Figure 35 these points represent the values of $H_K = \max_t \{|\rho^K(t)|\}$. The reported values are the $\rho^K(t)$.

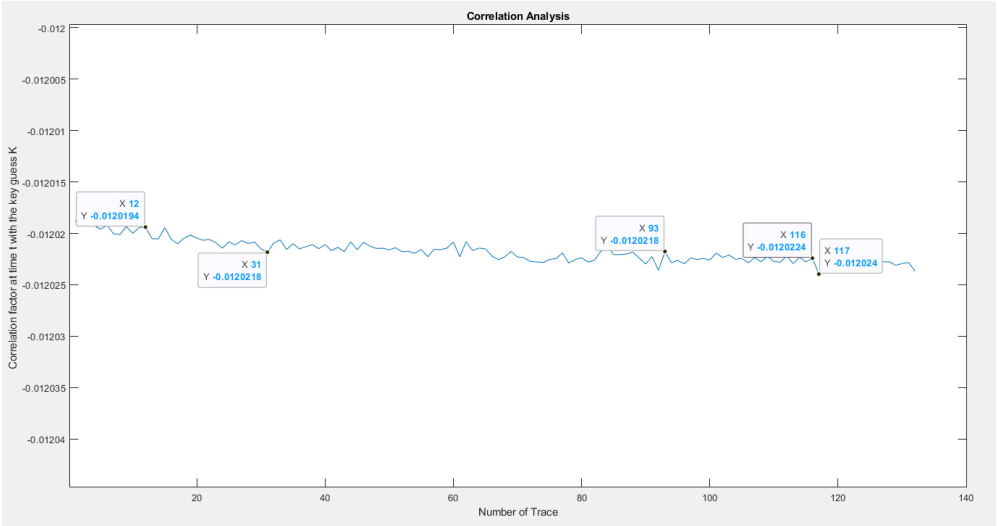


Figure 35. Correlation analysis plot with values of $\rho^K(t)$

# Trace	Magnitude (dB)	$\rho^K(t)$
12	-0.456	-0.01201
31	0.398	-0.01202
93	-0.491	-0.01202
116	-0.597	-0.01202
117	0.832	-0.01202

Table 4. Correlation factors at first and second load group

The values of the correlation factor (in Table 4) at time t are based on the key guess. This value is required to be used as a reading detector for the traces. It should be used to define how to discern from one trace to another one, and so discover how they are encrypted. In this presentation, it will be only reported, but in a study case, it should be placed in a collection to be used as a known value to recognize a Hamming weight measured $W_i^K = HW(S(p_i \oplus K))$ from another one. But this procedure should be done and presented as a separate work.

4.4 Differential frequency analysis

DFA, as discussed in Chapter 2, is an extension of differential analysis. The differential signals are computed in the frequency domain, this computation solves the problem of misalignment in traces because the FFT is time-shift invariant. Figure 36 uses the frequency $D_j^K(f)$ which is the difference between the positive and negative amplitudes, and H_j^K in Figure 36 is the plot that gathers them all.

$$D_j^K(f) = \frac{1}{\frac{n}{2}}\sum_i P_i^0(f) - \frac{1}{\frac{n}{2}}\sum_i P_i^1(f) \tag{2.6}$$

$$H_j^K = \sum_f D_j^K(f) \tag{2.7}$$

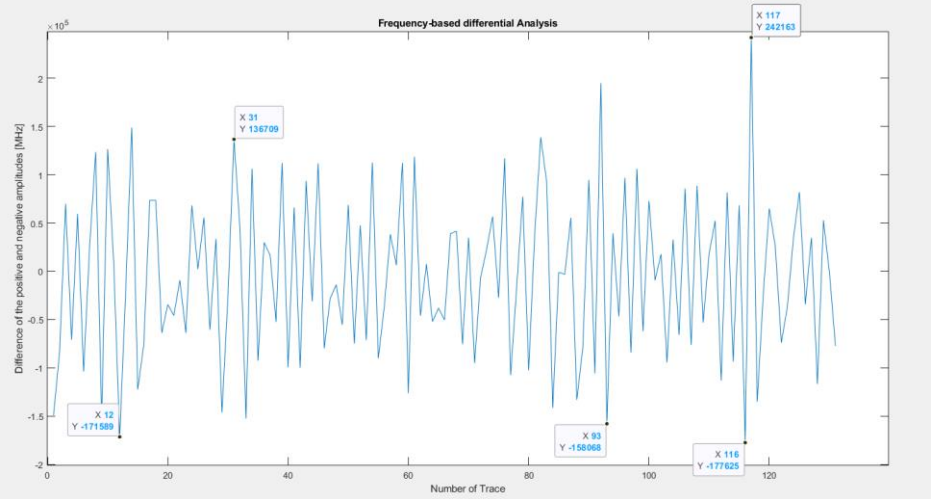


Figure 36. Differential frequency analysis plot of $D_j^K(f)$

In Figure 36 and Figure 37 is possible to observe the values in *MHz* at the point where the differential and correlational analysis are applied.

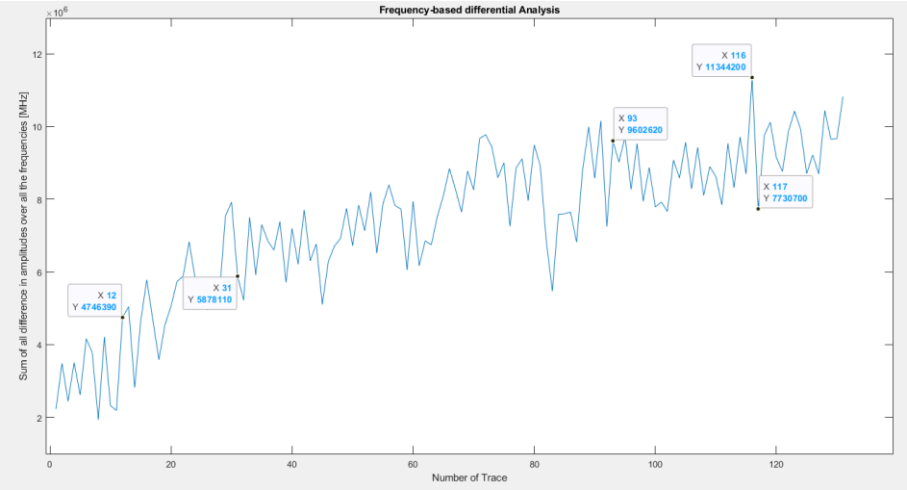


Figure 37. Differential frequency analysis plot $H_{jK}(f)$

From Figure 36 it should be possible to understand the trace misalignment associated with the acquisition of traces, because there is not any flat (or at very low oscillation) frequency interval at the end of the plot. Normally, this procedure is performed acquiring the EM traces in the inner part of the RoA, in order to study random selection and determine the time shift or delays. Frequency-based DEMA on a specific area defined as RoA with multiple sets of acquired EM traces using different keys can help to characterize the device, and so obtain the secret key of the device. This process should be repeated for each key byte of the key, until the 128-bit encryption is found.

# Trace	$D_j^K(f)$	$H_j^K(f)$
12	-0.171	4.746
31	0.136	5.876
93	-0.158	9.602
116	-0.242	11.344
117	0.177	7.730

Table 5. Significant differential trace $D_j^K(f)$ and heigh of the differential H_j^K at first and second load group.

The reported values in Table 5 are the intercepted values in differential analysis. From Figure 36 it is possible to affirm that the acquired values are correct because they continue to present intense amplitude. It is possible to add that the values from $D_j^K(f)$ and $H_j^K(f)$ can be used as a more enhanced tool to characterize the device, because thanks

to the ability of the being time-shift invariant, it is possible to enhance these studies under the signal processing theory, to find a correlation between acquired trace in frequency domain and key guess.

5. CONCLUSIONS

This work has presented an EM measurement bench for side-channel analysis made of components that are possible to find available from various sellers on the web. The bench allows to automate the process of discovering possible location emitting EM side channel leakage by scanning the DUTs surface and characterizing a possible region of attack. This region of attack can then be used to define the exact location where to position the EM probe, to perform further analysis (such as DEMA, CEMA, DFA), and to discover any SCA weaknesses, and where to apply the theory of the side channel analysis.

The device is based on a software-controlled EM leakage scan platform that can interconnect control and measurement devices at the same time, allowing all side channel leakage measurements to be performed, measured, and controlled in real time. An accurate H-probe for the detection of EM leakage and customized scanning software to perform SCA allow the system to be economical, versatile, and easy to implement according to the user's requests, without the need to have an expert in the field.

The measurement bench has the advantage of being able to define portions of the area to be scanned thanks to the possibility of launching multiple tests. `Test 1` consists in scanning a total area of the cryptographic processor, or a localized area, while `Test 2` scans a specific area from a certain position and therefore reduces measurement time costs. To get the shortest possible scanning time it is necessary to know the DUT architecture.

Furthermore, an algorithm to implement and perform side channel theory is implemented on MATLAB, which was needed to interface and link at the same time hardware and software, and it is a useful tool which provides a subsequent overview of the discovered data. In Arduino IDE, the implementation of the entire algorithm triggers the different Arduino boards, launches the Cryptographic Algorithm *AES,* and controls the measurement bench. Therefore, MATLAB controls Arduino and the oscilloscope and can launch different scripts to perform the three side-channel analysis techniques developed, that are Differential EM Analysis *(DEMA),* Correlational EM analysis *(CEMA)* and Differential frequency analysis (DFA). These analyses are implemented and provide the user with different useful tools to discover and study the SCA applications, but unfortunately, they require the users to understand the practical limitation and adapt the case to the machine. Even if this procedure is a strong way to discover the key, it requires that the user knows how to manipulate a cryptographic algorithm in a way to reduce the EM noise and

the different security protocol of the DUT. Thus, the three proposed theories require that the user has already studied the S-box temporal or frequency domain response, in order to organize an archive of the results provided in this work, that are the different combination plaintext/key-guess.

It is worth mentioning that the duration of the EM leakage measurement on the device is directly proportional to the increase in the size of the cryptographic chip to be analyzed; therefore, different techniques could be used. The results retrieved by the proposed work are consistent with the theory discussed in chapter 2. This project is implemented to be usable by an inexpert user that has basic knowledge in signal processing and electromagnetic theory. Nevertheless, this topic continues to present different challenges to be overcome by the user, not only from the point of view of the implementation but even from the point of view of the decision to be considered.

The decisions that require to be considered are the *region of attack* on the heatmap and the manipulation of the algorithm in order to reduce the EM noise. According to the chosen decisions, the implemented defense in the cryptographic algorithm will be reduced, and the cryptographic key will be exposed during `round 1` of `AddRoundKey` operation. Moreover, the implementation of a trigger can help the user by providing a way to reduce time-shift variance, increasing the performance of *DEMA* and *CEMA*.

However, the discussed uncertainties can be overcome with some knowledge from the user on the device, and the proposed work may be a useful addition to begin the study of a side channel analysis case.

REFERENCES

[1] C. H. Gebotys, *Security in Embedded Devices*. Springer US, 2010. doi: 10.1007/978-1-4419-1530-6.

[2] T. Moos, A. Moradi, and B. Richter, “Static Power Side-Channel Analysis - An Investigation of Measurement Factors,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 28, no. 2, pp. 376–389, 2020, doi: 10.1109/TVLSI.2019.2948141.

[3] K. Gandolfi, C. Mourtel, and F. Olivier, “Electromagnetic analysis: Concrete results,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2162, pp. 251–261, 2001, doi: 10.1007/3-540-44709-1_21.

[4] J. J. Quisquater and D. Samyde, “Electromagnetic analysis (Ema): Measures and countermeasures for smart cards,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2140, pp. 200–210, 2001, doi: 10.1007/3-540-45418-7_17.

[5] J. Danial, D. Das, S. Ghosh, A. Raychowdhury, and S. Sen, “SCNIFFER: Low-Cost, Automated, Efficient Electromagnetic Side-Channel Sniffing,” Aug. 2019, [Online]. Available: http://arxiv.org/abs/1908.09407

[6] S. Mangard, E. Oswald, and T. Popp, “Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security).”

[7] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, “ECDH key-extraction via low-bandwidth electromagnetic attacks on PCs,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9610, pp. 219–235, 2016, doi: 10.1007/978-3-319-29485-8_13.

[8] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, “Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9293, pp. 207–228, 2015, doi: 10.1007/978-3-662-48324-4_11.

[9] J. jaffe B. J. Paul Kocher, “Differential Power Analysis,” *Encycl. Cryptogr. Secur.*, pp. 336–338, 2011, doi: 10.1007/978-1-4419-5906-5_196.

[10] A. Matthews, “Low cost attacks on smart cards: the electromagnetic sidechannel,”

Next Gener. Secur. Softw. (September 2006), no. September, 2006, [Online]. Available: https://www.nccgroup.trust/media/18514/low_cost_attacks_on_smart_cards_the_electromagnetic_side_channel.pdf

[11] T. Kasper, D. Oswald, and C. Paar, “EM Side-Channel Attacks on Commercial Contactless Smartcards Using Low- Cost,” vol. 2908, no. January 2009, 2004, doi: 10.1007/978-3-642-10838-9.

[12] C. H. Gebotys, *Security in Embedded Devices*, no. Mills 2009. 2010. doi: 10.1007/978-1-4419-1530-6.

[13] A. P. Sayakkara and N. A. Le-Khac, “Forensic insights from smartphones through electromagnetic side-channel analysis,” *IEEE Access*, vol. 9, pp. 13237–13247, 2021, doi: 10.1109/ACCESS.2021.3051921.

[14] E. Ronen, C. O’Flynn, A. Shamir, and A. O. Weingarten, “IoT Goes Nuclear: Creating a ZigBee Chain Reaction,” *Proc. - IEEE Symp. Secur. Priv.*, pp. 195–212, 2017, doi: 10.1109/SP.2017.14.

[15] F. X. Standaert, E. Peeters, G. Rouvroy, and J. J. Quisquater, “An overview of power analysis attacks against field programmable gate arrays,” *Proc. IEEE*, vol. 94, no. 2, pp. 383–394, 2006, doi: 10.1109/JPROC.2005.862437.

[16] E. Tromer, “Hardware-Based Cryptanalysis,” 2007.

[17] D. Gerke and W. Kimmel, “Electromagnetic Compatibility 89.1 Grounding and Shielding in Industrial Electronic Systems Understanding EMI Problems • Grounding • Shielding 89.2 EMI and EMC Test Methods Nature of Electric and Magnetic Fields • Measurement Antennas • Measurement Enviro,” 1999.

[18] Langer-emv, “Langer EMV - ICR HH100-6 set, Near-Field Microprobe 2.5 MHz - 6 GHz,” 2020, [Online]. Available: https://www.langer-emv.com/en/product/near-field-microprobe-sets-icr-hh-h-field/26/icr-hh100-6-set-near-field-microprobe-2-5-mhz-6-ghz/758

[19] C. O. Flynn and Z. D. Chen, “ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research,” 2015.

[20] P. Technology, “Programmer’s Guide Flexible Resolution Oscilloscopes PicoScope ® 5000 Series (A API) I PicoScope 5000 Series (A API) Programmer’s Guide,” 2013.

[21] Mathworks, “PicoScope 5000 Series A API MATLAB Generic Instrument Driver -

File Exchange - MATLAB Central.” https://se.mathworks.com/matlabcentral/fileexchange/42820-picoscope-5000-series-a-api-matlab-generic-instrument-driver

[22] J. Heyszl, S. Mangard, B. Heinz, F. Stumpf, and G. Sigl, “Localized electromagnetic analysis of cryptographic implementations,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7178 LNCS, pp. 231–244, 2012, doi: 10.1007/978-3-642-27954-6_15.

[23] S. Mangard, “Hardware countermeasures against DPA - A statistical analysis of their effectiveness,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2964, pp. 222–235, 2004, doi: 10.1007/978-3-540-24660-2_18.

[24] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3156, pp. 16–29, 2004, doi: 10.1007/978-3-540-28632-5_2.

[25] E. Brier, C. Clavier, and F. Olivier, “Correlation Power Analysis with a Leakage Model.”

[26] R. M. Kosanke, “Forensic Insights From Smartphones Through Electromagnetic Side-Channel Analysis,” *Secur. Embed. Devices*, pp. 163–222, 2019, doi: 10.1007/978-1-4419-1530-6 8.

[27] Wikipedia, “Kerckhoffs’s principle.” https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle

[28] Catherine H. Gebotys, “Security in Embedded Devices.,” in *Security in Embedded Devices.*, Waterloo Canada: Springer., 2009, pp. 163–222.

[29] C. H. Gebotys, S. Ho, and C. C. Tiu, “EM analysis of Rijndael and ECC on a wireless Java-based PDA,” *Lect. Notes Comput. Sci.*, vol. 3659, pp. 250–264, 2005, doi: 10.1007/11545262_19.

[30] Y. Liu and B. Ravelo, “Fully time-domain scanning of EM near-field radiated by RF circuits,” *Prog. Electromagn. Res. B*, vol. 57, no. 57, pp. 21–46, 2014, doi: 10.2528/PIERB13072903.

[31] J. Danial, S. Member, D. Das, S. Member, and S. Ghosh, “SCNIFFER : Low-Cost , Automated , Efficient Electromagnetic Side-Channel Sniffing,” pp. 1–12.

[32] Mathworks, “Connection to serial port - MATLAB - MathWorks Nordic,” *R2019b*.

https://se.mathworks.com/help/matlab/ref/serialport.html

[33] D. Das, M. Nath, B. Chatterjee, S. Ghosh, and S. Sen, “STELLAR: A Generic em Side-Channel attack protection through ground-up root-cause analysis,” *Proc. 2019 IEEE Int. Symp. Hardw. Oriented Secur. Trust. HOST 2019*, pp. 11–20, 2019, doi: 10.1109/HST.2019.8740839.

[34] V. Carlier, H. Chabanne, E. Dottax, and H. Pelletier, “Electromagnetic Side Channels of an FPGA Implementation of AES.”

[35] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, “The em Side-Channel(s),” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2523, pp. 29–45, 2003, doi: 10.1007/3-540-36400-5_4.

[36] C. Ramsay and J. Lohuis, “Covertly stealing keys for $ 200 Introduction,” 2017.

[37] A. M. Qazi, S. H. Mahmood, A. Haleem, S. Bahl, M. Javaid, and K. Gopal, “The impact of smart materials, digital twins (DTs) and Internet of things (IoT) in an industry 4.0 integrated automation industry,” *Mater. Today Proc.*, vol. 62, pp. 18–25, 2022, doi: 10.1016/j.matpr.2022.01.387.

[38] B. CARSTEN, “Sniffer probe locates sources of EMI,” *Edn*, pp. 1–7, 2012, [Online]. Available: http://jahonen.kapsi.fi/Electronics/Stuff/EMI_Sniffer_EDNMAG.pdf

[39] DFRobot, “Stepper_Motor_Shield_For_Arduino_DRV8825__SKU_DRI0023-DFRobot.” https://wiki.dfrobot.com/Stepper_Motor_Shield_For_Arduino_DRV8825__SKU_DRI0023

[40] T. Schneider and A. Moradi, “Leakage Assessment Methodology,” pp. 495–513, 2015, doi: 10.1007/978-3-662-48324-4_25.

[41] D. B. Roy, S. Bhasin, S. Guilley, A. Heuser, S. Patranabis, and D. Mukhopadhyay, “CC Meets FIPS: A Hybrid Test Methodology for First Order Side Channel Analysis,” *IEEE Trans. Comput.*, vol. 68, no. 3, pp. 347–361, 2019, doi: 10.1109/TC.2018.2875746.

[42] V. Lomné *et al.*, “Modeling Time Domain Magnetic Emissions of ICs To cite this version : HAL Id : lirmm-00762033 Magnetic Emissions of ICs,” 2019.

[43] L. Sauvage *et al.*, “ElectroMagnetic Radiations of FPGAs : High Spatial Resolution Cartography and Attack of a Cryptographic Module To cite this version : HAL Id :

hal-00319164 ElectroMagnetic Radiations of FPGAs : High Spatial Resolution Cartography and Attack of a Cryptogra," 2008.

[44] lastminuteengineers, "In-Depth: Interface DRV8825 Stepper Motor Driver Module with Arduino." https://lastminuteengineers.com/drv8825-stepper-motor-driver-arduino-tutorial/

[45] Mike McCauley (mikem@airspayce.com), "AccelStepper: AccelStepper library for Arduino", [Online]. Available: http://www.airspayce.com/mikem/arduino/AccelStepper/

[46] MathWorks, "Write line of ASCII data to serial port - MATLAB writeline." https://se.mathworks.com/help/instrument/serialport.writeline.html

[47] airspayce, "AccelStepper: AccelStepper Class Reference." http://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html

[48] kokke, "GitHub - kokke/tiny-AES-c: Small portable AES128/192/256 in C", [Online]. Available: https://github.com/kokke/tiny-AES-c#readme

[49] MathWorks, "Read line of ASCII string data from serial port - MATLAB readline - MathWorks Nordic," *R2019b*. https://se.mathworks.com/help/instrument/serialport.readline.html

[50] MathWorks, "Signal-to-noise ratio - MATLAB snr - MathWorks Nordic", [Online]. Available: https://se.mathworks.com/help/signal/ref/snr.html?searchHighlight=sNR &s_tid=srchtitle_sNR _1

[51] MathWorks, "Create heatmap chart - MATLAB heatmap - MathWorks Nordic." https://se.mathworks.com/help/matlab/ref/heatmap.html?searchHighlight=Heatmap&s_tid=srchtitle_Heatmap_1

[52] MathWorks, "Fast Fourier transform - MATLAB fft." https://se.mathworks.com/help/matlab/ref/fft.html

[53] Mathworks, "Absolute value and complex magnitude - MATLAB abs - MathWorks Nordic." https://se.mathworks.com/help/matlab/ref/abs.html