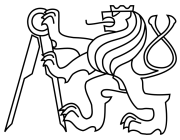
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF BACHELOR'S THESIS

Title:	3D near Field Acquisition System
Student:	Klára Dvořáková
Supervisor:	Ing. Jiří Buček
Study Programme:	Informatics
Study Branch:	Information Technology
Department:	Department of Computer Systems
Validity:	Until the end of summer semester 2017/18

Instructions

Create a system for mapping of electromagnetic emanation of secret information from electronic circuits. Build a device that will position a measuring probe in space and analyze the measured signals with respect to values processed in a digital circuit (a smart card). Use a 3D printer mechanics as a base for positioning and a digital oscilloscope for the measurement.

References

Will be provided by the supervisor.

prof. Ing. Róbert Lórencz, CSc.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague January 29, 2017

Czech Technical University in Prague
Faculty of Information Technology
Department of computer systems

Bachelor's thesis

3D Near Field Acquisition System

Klára Dvořáková

Supervisor: Ing. Jiří Buček

10th May 2018

Acknowledgements

First of all, I would like to thank my supervisor Ing. Jiří Buček for his patience and valuable advice. Thanks to my friends and my family who supported me during my studies. Moreover, thanks to all the people who helped me with this theses and who were the psychological support to me during these difficult times.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 10th May 2018 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Dvořáková, Klára. *3D Near Field Acquisition System.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstrakt

Tato práce se zabývá přestavením 3D tiskárny na přístroj umožňující automaticky snímat elektromagnetický postranní kanál vycházející z mikroprocesoru (čipové karty). Cílem je na přístroji signál pomocí digitálního osciloskopu naměřit a sestavit 3D graf, který bude zobrazovat množství vyzářeného signálu v jednotlivých místech mikroprocesoru. Literární rešerše se zabývá elektromagnetickým postranním kanálem a způsoby analýzy naměřených dat, zejména pak diferenciální elektromagnetickou analýzou. Praktická část práce navazuje sestavením přístroje na automatizované měření, dále pak programem, který řeší komunikaci mezi počítačem, osciloskopem, pozicovacím přístrojem a čipovou kartou. Závěrečná část práce je věnována analýze a dešifrování naměřených dat. Výsledek práce nabízí levnou alternativu pro již existující drahé přístroje měřící postranní elektromagnetické záření.

Klíčová slova Postranní kanály, elektromagnetické záření, diferenciální elektromagnetická analýza, 3D tiskárna, blízké pole, smart card, AES

Abstract

This thesis describes rebuilding a 3D printer into a device that automatically captures electromagnetic side channel of a microprocessor (smart card). The aim is to measure the signal emitted by a smart card using a digital oscilloscope and to plot a 3D graph displaying the amount of signal radiated by each part of the smart card. The theoretical part of this paper deals with electromagnetic side channel and methods of analysis of measured data, particularly with differential electromagnetic analysis. The practical part of the paper follows with building a positioning device for automated measurements, then with a program that handles communication between a computer, the oscilloscope, the positioning device and the smart card. Finally, the program analyses the measured data. The resulting device offers a cheap alternative to already existing expensive devices measuring electromagnetic emissions.

Keywords Side channels, electromagnetic emission, differential electromagnetic analysis, 3D printer, near field, smart card, AES

Contents

List of Figures

List of Tables

Introduction

The development of computer and communication systems is getting faster and with this development new attacks are created. Using more sophisticated attacks means higher demands on cryptographic systems. Cryptanalysis focused on mathematical attacks can seem inefficient these days. Novel types of cryptanalysis are created. Attacks which deal with side channels and are focused on implementation of an algorithm or a protocol can be an example of this kind of attack. They do not primarily try to find a weakness in an algorithm, but they tend to misuse any information they can get from the encrypting device.

This paper deals with electromagnetic side channel. Various parts of cryptographic modules emit different amounts of electromagnetic emission. The aim is to build a positioning device that can automatically measure electromagnetic radiation and process measured data. We will measure the signal in different positions with respect to a smart card's chip.

In the theoretical part, we will focus on a description of the side channels, especially electromagnetic side channel. Then we will describe attacks using electromagnetic side channels, such as simple electromagnetic analysis or differential electromagnetic analysis. We will talk about ciphers and encryption with the emphasis on AES cipher. Then we will describe devices used for building our working environment and for the practical part of the thesis.

In the practical part, firstly we will focus on the realisation of the working environment. Then we will implement a program that can communicate with devices used in the working environment, position a probe, save signal measured by an oscilloscope and that can process measured data. Finally, we will test this program and measure electromagnetic radiation in different positions, process data, try to calculate correct cipher key using differential electromagnetic analysis and make graphs out of the results.

Chapter 1

Ciphers and Attacks

Codes and ciphers have been used for a few thousand years now to disguise the meaning of secret messages, from the simplest ciphers of ancient times to the modern encryption methods used today. Nowadays, ciphers are a necessary part of computer's communication. New ciphers are made to ensure that an attacker cannot eavesdrop on someone's conversation or steal someone's data. These ciphers are built to be mathematically almost impossible to break, or it would take such a long time to reveal them, and it would not be relevant for an attacker anymore.

1.1 Side-Channels

Cryptanalysis of side-channels is a relatively recent type of attack. These types of attacks are focused on implementation of the cipher rather than its mathematical principle. This characteristic makes them applicable even to ciphers that are almost impossible to break mathematically.

In an ideal world, information gets to a device, it is processed and the output is generated without any information leakage. To guarantee there is no side channel leakage, this device would have to draw constant current, have perfect electromagnetic screening shield etc. We can consider each undesirable type of communication as a side-channel. In real-world applications, every cryptographic module consumes power and emits electromagnetic or heat radiation depending on processed data. These channels can reveal sensitive data to the outside world.

Types of side-channels:

1. Power consumption
2. Timing

3. Electromagnetic radiation
4. Optical
5. Fault
6. Acoustic channel

1.1.1 Side-Channel Attacks and Countermeasures

Side-channel attacks can be both invasive and non-invasive. When conducting an invasive attack, an attacker needs to manipulate physically with a device they want to attack. Most of the side-channel attacks are non-invasive, they do not leave tracks, and they do not destroy or damage any data or system. We do not have to be in possession of the device, and the victim might not know we are attacking their system.

History

Military organisations were the first who used side-channel attacks. They created TEMPEST standards [6], [7], [8] which determine required computer protection against these attacks but also deal with attacking other systems. Some of these standards were later revealed to the public.

In the public area, a scientist from the Netherlands, van Eck, was the first to describe and conduct this attack [9]. He managed to measure the size of the radiation of computer monitors and extracted original image from this data. Scientists Kuhl and Anderson followed up with him and built a special cling film which protected the screen from the attack. Agrawal, Archambeault, Rao and Rohatgi followed on TEMPEST in their book [10], where they managed to prove that electromagnetic side-channel attacks are realisable and sometimes have better results than until that time used power side-channel attacks.

Description of an attack

As we can see in picture 1.1, plaintext comes to the cryptosystem and is encrypted with a cipher, during the encryption there is an information leakage caused by side-channels.

The advantage of these types of attacks is that most of the devices are not prepared for side-channel attacks. Nowadays, countermeasures are starting to take place. One of the possibilities is to add unrelated operations during encryption or decryption; this can help mostly against power and electromagnetic side-channel attacks. It will add noise to consumption or electromagnetic emissions and make it harder for an attacker to distinguish noise from a signal caused by encrypting or decrypting operations. Another protection against

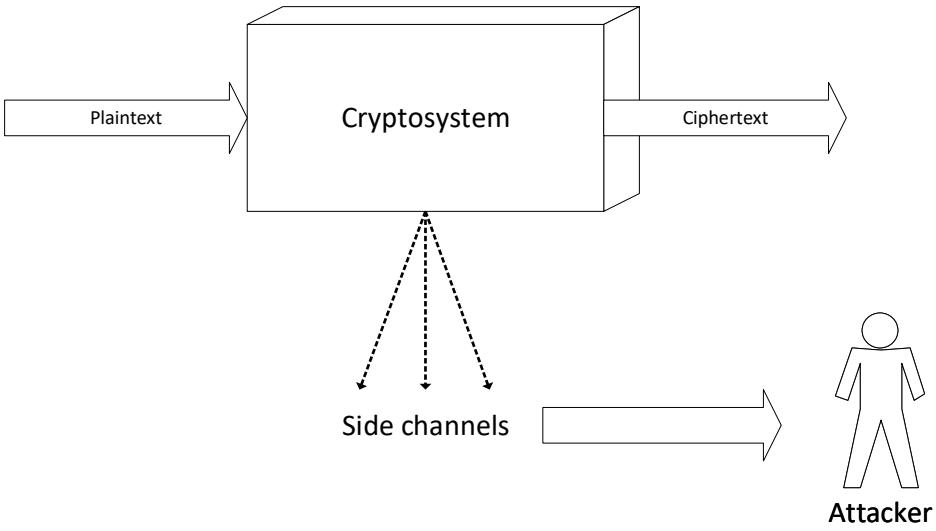


Figure 1.1: Side-channels

electromagnetic side-channel is an electromagnetic shield, which is nowadays commonly used to protect payment cards. There are more types of protections, but these are the most prevalent.

1.1.2 Electromagnetic Side-Channel

This section describes electromagnetic side-channel; more information can be found at [11]. Electromagnetic side-channel attack is based on power consumption side-channel attack. It is subject to the law of electromagnetic field, which was created by James Clerk Maxwell, and its four fundamental mathematical equations [12]. Based on his laws, we can say that each change of power or voltage causes a change of electromagnetic field. We can describe this realationship using Lenz's law [13]. This law is based on the one of the Maxwell's laws, Faraday law.

$$\mathcal{E} = \frac{\partial \delta \sigma}{\partial \delta t} \qquad \textit{(Lenz's Law)} \tag{1.1}$$

At Equation 1.1, $\mathcal{E}$ stands for electromotive force and δ represents magnetic flux. It describes the relationship between the change of electromagnetic field and the change of power or voltage.

The crucial difference between power and electromagnetic side-channel is that power consumption is a simple amplitude waveform over time, while the electromagnetic emission is a three-dimensional vector field that changes over time. To describe it, we need to use vector calculus techniques.

Electromagnetic emissions can be both intentional and unintentional. For example, an antenna emits electromagnetic emissions intentionally. All RFID enabled cards have an antenna connected to their chip for communication with the outside world. Another example can be a crystal oscillator defining the exact frequency of an internal clock. Power consumption of circuit elements can cause unintentional electromagnetic emissions. Through the Maxwell laws, the electromagnetic field exists around them. Unintentional electromagnetic emissions can be then used to attack the device.

We can use electric or magnetic probes to measure the electromagnetic field. Small probes can be used to measure near-field, the closer to the source, the more accurate the results are. When measuring near-field, the maximal distance should be the wavelength from the source.

1.1.2.1 Electromagnetic zones

The behaviour of small electromagnetic sources or antennas can be simplified and divided into two approximations: near field and far field. Consider a sinusoidal signal f going from point A to point B, where the distance between these two points is r. The signal travelling in a medium with relative dielectric constant ϵ_r and relative permeability μ_r propagates with speed $v = \frac{c}{\sqrt{\epsilon_r \mu_r}}$, where c is a constant, which represents speed of electromagnetic waves in vacuum, of value $2.998 \cdot 10^8$ m/s. We can speak about a "small" distance if at time $t = \frac{k}{v}$, where t is time needed for a variation at point A to travel to point B, the same variation can still be at point A. This is true, when $t \ll T = \frac{1}{f}$. In practice, we do not need to measure the time, we can use the wavelength λ, because $\lambda f = v$, we can say that $r \ll \lambda$.

Near field is then defined as follows

$$kr \ll 1 \quad or \quad r \ll \frac{\lambda}{2\pi} \qquad \text{(near field)}$$

$$r \gg \frac{\lambda}{2\pi} \quad or \quad kr \gg 1 \qquad \text{(far field)}$$

The electromagnetic signal in near field has high current and low voltage. The source is mainly magnetic. Nature of the wave depends on the source's characteristics, while the wave measured in far-field depends on the propagation medium. Electric and magnetic fields are both planar waves. To see the border between near field and far field, it is necessary to know the wavelength distance λ. At a wavelength distance away from the source, the impedance of all waves approaches the impedance of the free space, which is 377 Ω.

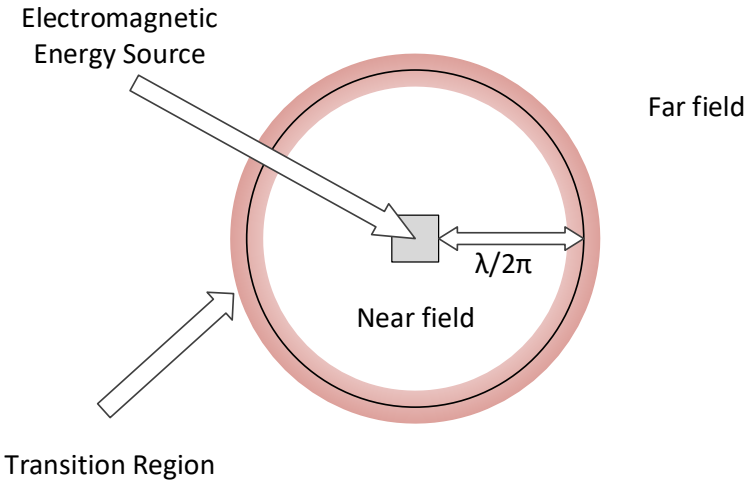


Figure 1.2: Electromagnetic zones

1.1.2.2 Electromagnetic exploitation

There are more approaches to exploiting electromagnetic emission. [11]

1. **Direct Local Radiation**
   This approach is technically the most difficult of the presented approaches. The amplitude is minimal and therefore a sensitive sensor is needed. Its best use is for measuring magnetic field when rapid changes in the current consumption generate the field.

2. **Modulated Signals**
   Signals can become modulated on top of stronger carriers. The modulated signals are measurable at a larger distance.

3. **Remote Electromagnetic Analysis**
   There are two types of this approach. First one is near-field, which we will use in this paper to measure electromagnetic emission. The second one is far-field.

4. **Remote Power Analysis**
   This approach is the same as electromagnetic analysis, but instead of the radiation originating from the power consumption on the chip, the field to power up RFID cards is analysed.

1.2 AES

AES stands for Advanced Encryption Standard. FIPS 197 [14] fully describes this cipher. It is a symmetric block cipher, i.e. it is using the same key for encrypting and decrypting, and it divides the plaintext into blocks of a given length, precisely 128 bits.

It is considered to be a secure cipher and a standard for encrypting information. It is commonly used for data encryption. The only successful attack at this cipher was a side-channel attack. It substituted previously used DES cipher that is less secure. Appendix A consists of a description of this cipher.

Chapter 2

Electromagnetic Analysis

Analysis of a side-channel can be both invasive and non-invasive. An electromagnetic analysis is typically non-invasive attack. Attacks are conducted to detect sensitive data from victim device.

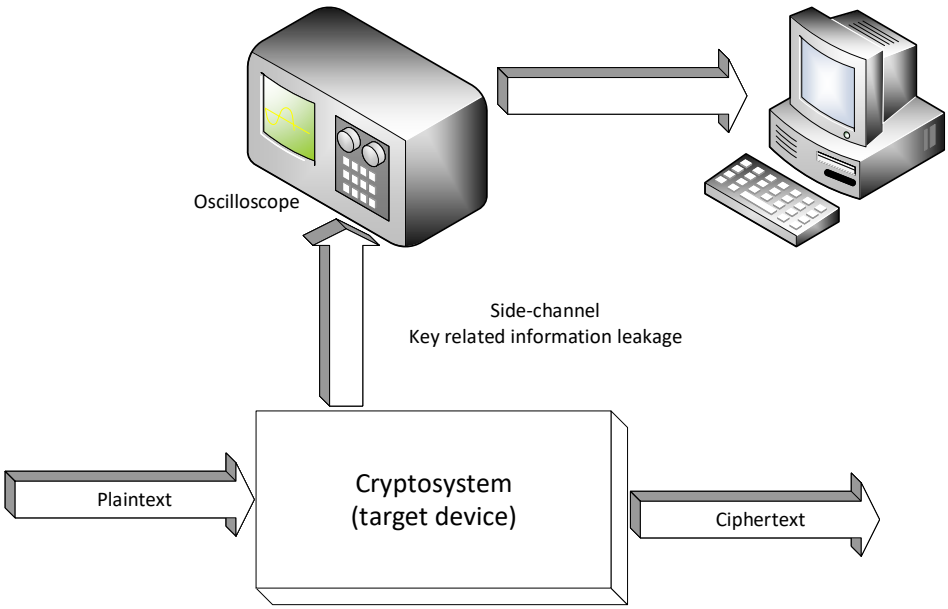


Figure 2.1: Electromagnetic analysis

As you can see in Figure 2.1, typical setup consists of a target device, where ciphers like DES or AES can be implemented, an oscilloscope and a computational platform.

Plaintext is sent to the target device, where encryption takes place, and ciphertext goes out of the device. While encrypting data, the oscilloscope measures electromagnetic radiation via probes, and the resulting measurements are sent to the computational platform, where an attacker can take an advantage of them.

There are two basic types of electromagnetic analysis: Simple electromagnetic analysis and Differential electromagnetic analysis.

2.1 Simple Electromagnetic Analysis (SEMA)

Simple electromagnetic analysis requires just one measurement or only a few of them. It is very similar to simple power analysis (SPA), described in [15]. It makes use of a relation between an instruction that is being executed and information which is gathered from an electromagnetic side-channel. It is, therefore, a direct data processing. An attacker is directly dealing with electromagnetic radiation and its changes in an attacked device at a specific time.

It is essential to have detailed knowledge of the algorithm implemented on the victim's machine and to have knowledge about the cryptographic device itself. It is also crucial to have a strong relationship between electromagnetic radiation and the value of the secret key.

Unlike simple or differential power analysis, which can measure power consumption by measuring ends of a power source, SEMA needs a special probe (antenna) for precise measurements.

There are two types of simple electromagnetic analysis. First one is single-shot SEMA, which needs only one measurement. The second one is multiple-shot SEMA; this type requires more measurements than just one. Multiple measurements can eliminate some unwanted noise. We can measure electromagnetic radiation for either the same plaintext each time or a different one.

2.2 Differential Electromagnetic Analysis (DEMA)

DEMA is very similar to DPA (Differential Power Analysis), which is described in [15]. In contrary to SEMA, DEMA needs a lot of measurements. The number of measurements ranges in thousands. Since there are more measurements taken, there is a higher possibility to reduce noise and some wrong values from the measured data. Compared to SEMA, DEMA is much more efficient. There is also no need for detailed knowledge of the cryptographic

system and the implemented algorithm as it was in SEMA.

Data processing is different as well. In SEMA, measured data were processed directly at a specific time to find a template or a pattern. DEMA does not need any waveform at a specific timeline; it leverages a relation between constant time and processed value. Another advantage of DEMA is that it guesses a small part of a secret key at a time. For instance, DEMA can guess a single byte at a time, which makes it possible to try all the values of the key. It only tries 2^8 possible values for each byte of the key. Taking 16-byte key, the algorithm has to try $16 \cdot 2^8 = 2^{12}$ possibilities instead of 2^{128}. The whole diagram of DEMA can bee seen in Figure 2.2.

There are five steps in DEMA:

1. Choose intermediate result.
2. Measure electromagnetic field.
3. Calculate the hypothetical intermediate result.
4. Map hypothetical intermediate results to measured data.
5. Compare hypothetical values to measured data.

Used abbreviations

- $f(d,k)$ ... hypothesis function for calculating intermediate result
- D ... number of input data blocks, number of measurements
- $\mathbf{d}$ ... vector with input data for one measurement (one trace)
- T ... length of the trace
- $\mathbf{T}$ ... matrix of size D×T, measured traces
- k ... key candidate
- $\mathbf{k}$ ... vector with all the possible keys, key hypotheses
- K ... total number of possible key choices
- $\mathbf{V}$ ... matrix of size D×K, intermediate result
- $v_{i,n}$ ... element of matrix $\mathbf{V}$
- v_{ck} ... vector, values for the right cipher key
- k_{ck} ... secret key

- **H** ... matrix of hypothetical values of electromagnetic emissions
- **R** ... matrix of size KxT, comparision between H and T
- *ct* ... index of measured data that are dependent on intermediate result
- *ck* ... index of the right cipher key
- *i* ... index of input data, 1,..., D
- *j* ... index of traces, 1,..., T
- *n* ... index of key candidates, 1,..., K

2.2.1 Choose Intermediate Result

Intermediate value is chosen according to the hypothesis function $f(d, k)$, where d is (in most cases) a plaintext or a ciphertext sent to the device and k is a subkey. Hypothesis f is determined by the encryption algorithm as well; it is a specific function of the implemented algorithm. It must be a function of known variable d and a small part of a key k. According to that, one part is known and the other one can be guessed.

We need the intermediate value for estimation of the correlation between electromagnetic emission and the internal data or operations performed during the computation of the cipher. For example, we can assume that the intermediate value with more bits set to 1 will emit more electromagnetic signal. We will use the intermediate result to decide whether the correlation exists or not in subsection 2.2.6.

2.2.2 Measure Electromagnetic Field

Second step is to measure electromagnetic field of cryptographic device when encrypting or decrypting various data D. An attacker needs to know vector (or they can choose it) **d**, where $\mathbf{d} = (d_1, \ldots, d_D)$' with input data for each measurement. Measurements of electromagnetic field $(t_{i,1}, ..., t_{i,T})$ correspond with data d_i. We conduct a measurement for each of the D data blocks. Data must be assigned to its measurement and saved. They are used in the first step to compute the intermediate result.

The result of the measurement is a matrix **T** of size D×T. For each input, we save the same part of the measurement. The same operation must correspond to each column t_j. The best way to obtain this is to use a trigger signal to record the electromagnetic emissions of the same sequence of operations for each encryption or decryption run.

2.2.3 Calculate the Hypothetical Intermediate Results

In this step, hypothetical intermediate results are calculated for each possibility of chosen part of the key $\mathbf{k} = (k_1, \ldots, k_K)$, where $k_1, \ldots, k_K$ are hypothetical values of the key and K is the maximum number of hypothetical keys. Intermediate value $f(d, k)$ is calculated for all D encrypting or decrypting operations and all K key hypotheses. The result is a matrix $\mathbf{V}$ of size D×K.

$$v_{i,n} = f(d_i, k_n) \tag{2.1}$$

Each column of matrix $\mathbf{V}$ consists of results computed from the hypothesis of key k_j, which also means the correct cipher key used in the device. The goal is to find the column with values which were processed during encrypting or decrypting operations in the device. When we find this column, we immediately know the secret key k_{ck}.

2.2.4 Map Intermediate Results to Hypothetical Values of Electromagnetic Emissions

Hypothetical intermediate values from matrix $\mathbf{V}$ are mapped to hypothetical electromagnetic emissions in matrix $\mathbf{H}$. The electromagnetic emission is estimated according to a defined model for each value $v_{i,n}$ in order to obtain a hypothetical electromagnetic emission value $h_{i,n}$. Accuracy depends on the attacker's knowledge of the device. Hamming weight is a commonly used model; it is the easiest way, and the attacker does not need to have any further knowledge about the inner structure of the device.

2.2.5 Compare Hypothetical Values to Measured Data

Result of this step is a matrix $\mathbf{R}$ of size K×T, where each element $r_{n,j}$ is a result of comparison between each column h_n and t_j. Determination of the key takes advantage of the fact that electromagnetic waveforms correspond to the electromagnetic emissions in the device during encryption or decryption for different input data. Intermediate result chosen in the first step must be a part of the algorithm. As a result of that, some parts in a specific time are dependent on intermediate result. This part of the measured data can be called *ct*. T_{ct} consists of electromagnetic emission, which depends on intermediate result v_{ck}.

Hypothetical values h_{ck} were estimated based on v_{ck} values. Columns h_{ck} and t_{ct} are strongly dependent on each other. They are leading to the highest absolute value in the matrix $\mathbf{R}$. The highest value of $\mathbf{R}$ is $k_{ck,ct}$. Other values are lower because they are not that much dependent on each other. Index for the right key is the highest value *ck* and the highest value of time moment *ct*. Indexes of these two values are the results of DEMA.

2.2.6 Attack Based on Correlation Coefficient

Pearson correlation coefficient is commonly used to establish a linear dependency between two random variables. Correlation is defined for two quantities X and Y using covariation by the relation:

$$\rho = \frac{Cov(X,Y)}{\sqrt{\sigma^2(X)\sigma^2(Y)}}. \tag{2.2}$$

Covariation Cov(x,y) is the expected value of the product of deviations of the random variables from their individual expected values.

$$cov(X,Y) = E[(X - E[X])(Y - E[Y])], \tag{2.3}$$

where sigma is the standard deviation of the random variable. The values of the correlation coefficient range from -1 to 1. The value of 1 implies that there is a linear relationship between X and Y. A 0 value implies that there is no linear correlation between these two variables and -1 value implies that there is inverse linear relationship.

Usually, we do not know the precise distributions of X and Y, because we can perform a limited amount of measurement. If we have a sample of values of X and Y, we can compute the sample correlation coefficient:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.4}$$

Equation according to DEMA matrixes:

$$r_{i,j} = \frac{\sum_{i'=1}^{D}(h_{i',i} - \bar{h_i})(t_{i',j} - \bar{t_j})}{\sqrt{\sum_{i'=1}^{D}(h_{i',i} - \bar{h_i})^2 \sum_{i'=1}^{D}(t_{i',j} - \bar{t_j})^2}} \tag{2.5}$$

The correlation coefficient is not the only way of comparing data. The difference or the distance of means can be also used. When the attacker has better knowledge of the device, they can use a template-based attack. For more information on template attacks, see for example [15], but in subsequent text, we will use the Pearson correlation coefficient only.

2.3 DEMA on AES

The main step in the differential electromagnetic analysis is to select a hypothesis *f*. Selection differs according to an encryption algorithm, but it must be a function of a known variable d and a secret key k. In this paper, we will use output of function *AddRoundKey()* in the initialisation part of the algorithm, which is the input for the substitution table S-Box. The function is called *SubBytes()* transformation, which is a non-linear byte substitution that operates independently with S-Box, see Appendix A.

$$v_{i,n} = f(d_i, k_n) = SBox(PT_{i,m} \oplus k_n), \tag{2.6}$$

where $d_i = PT_{i,m}$ is the m^{th} byte of the i^{th} block of the plain text PT, and k_n is a hypothetical key that guesses the value of the right key byte K_m.

It does not matter which output bit of S-Box output is used as a function f; it always corresponds not only to the bit in $PT_{i,m}$ or K_n, but also to the changes of other input bits. This ensures relative independence of f values calculated using right and wrong key. Because of that, it reduces correlation between the measured data and the predicted values calculated based on the wrong key. This makes it easier to identify the right key. [16]

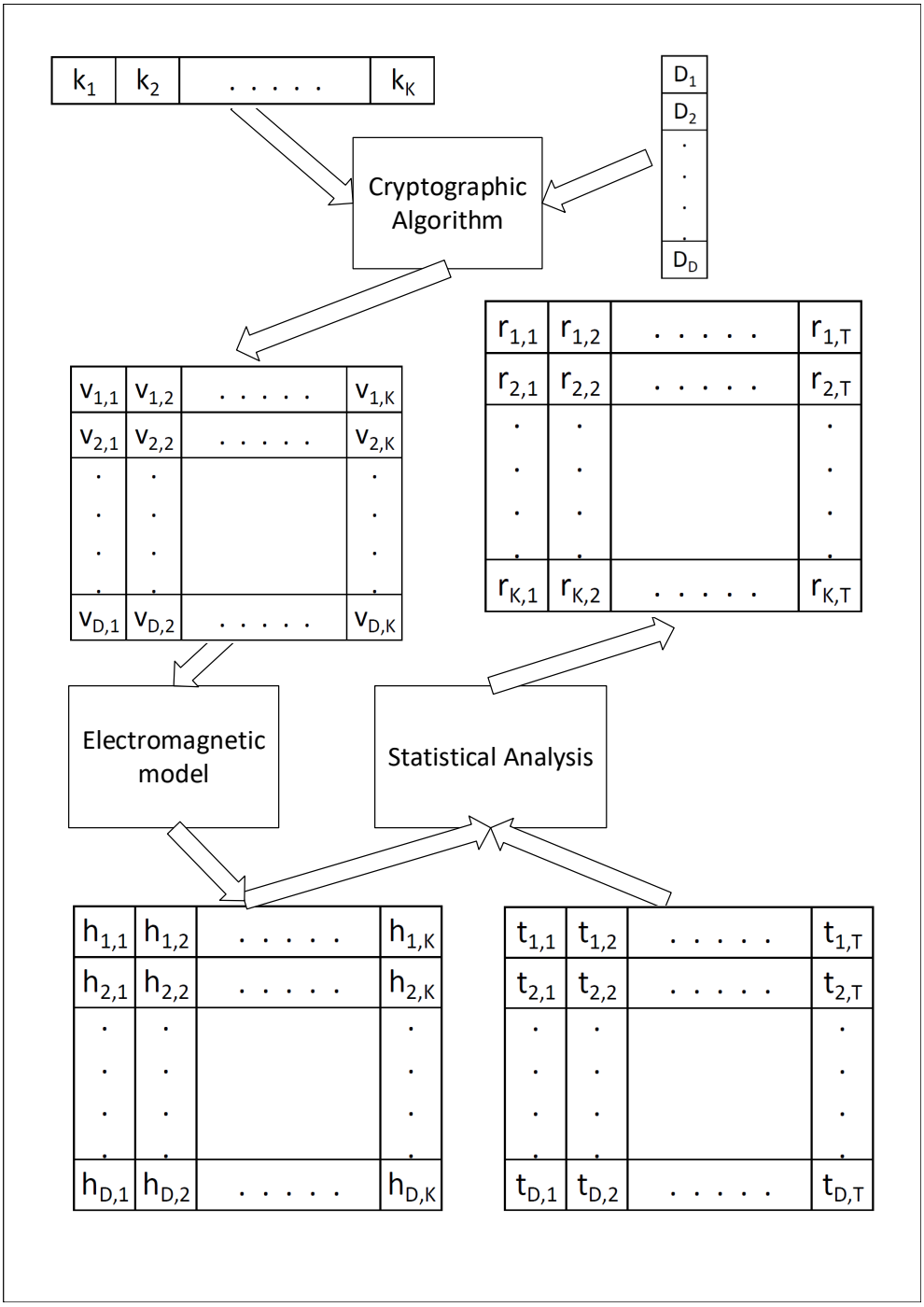


Figure 2.2: Differential electromagnetical analysis

Chapter 3

Near-Field Cartography

Near-field cartography is essential for finding the hot spot of a measured device. An electromagnetic probe is small compared to the measured device; it is necessary to find the place with the highest signal. It can be done by manual search, where we can look for the highest value of the electromagnetic emission, but it is neither the smartest nor the most efficient solution.

The best way to do that is to build a device, which will move the probe automatically. We can use more techniques to do that. There are several papers employing a positioning device for moving probe and measuring electromagnetic emissions. However, there is no further description of the equipment used for holding and moving the probe during the measurement.

There are also more approaches how to decide the best place to attack. This chapter is focused on these methods.

3.1 Indicators of the best spot to attack

Several techniques can be used to calculate the most interesting spots on a measured device. The basic approaches how to get the best spot to attack are described in [17].

The first approach is to take the maximum of each trace and to compare the ratio between the noise and the interesting trace. It needs three measurements, one for the encryption/decryption and two for the noise. Then the SNI_{max} (maximal difference indicator) is defined as:

$$SNI_{max} = \frac{MAX_0^T(|\Gamma_{AES} - \Gamma_{noise}|)}{MAX_0^T(|\Gamma_{noise2} - \Gamma_{noise}|)} \tag{3.1}$$

Γ_{AES} represents the measured signal where the source was AES and Γ_{noise2}

is measured noise.

The best place to attack is where SNI_{max} has the highest value, due to the increase of the signal level during the encryption/decryption. Instead of SNI_{max}, we can also use time-integrated indicator SNI_t, which can represent the consumption of the encryption/decryption. SNI_t can be calculated according to this equation:

$$SNI_t = \frac{\int_0^T (|\Gamma_{AES} - \Gamma_{noise}|)}{\int_0^T (|\Gamma_{noise2} - \Gamma_{noise}|)} \tag{3.2}$$

However, the time-integrated indicator SNI_t is less precise than SNI_{max}.

The second approach is an S-Box maximal difference indicator. Firstly, we must find a position, where the signal correlates with the AES S-Box the most. This indicator tries to remove logical noise that is caused by the other bits activity. Targeted logical object is the output of the S-Box i, where $0 \leq i \leq 16$. The main source of the logical noise is assumed to be the other S-Boxes activity. In this approach, we must choose a plaintext with a variable part and a fixed part. Variable part stimulates the activity of the i^{th} S-Box, while the fixed part ensures that the activity of other S-Boxes is kept constant. It also has two ways of calculating the best position to attack. First one is MDI_{max}, which is defined as:

$$MDi_{max} = \frac{MAX_0^T (|\Gamma_{M1} - \Gamma_{M2}|)}{MAX_0^T (|\Gamma_{M1'} - \Gamma_{M1}|)} \tag{3.3}$$

The second one is also the time-integrated way MDI_t defined as:

$$MDI_t = \frac{\int_0^T (|\Gamma_{M1} - \Gamma_{M2}|)}{\int_0^T (|\Gamma_{M1'} - \Gamma_{M1}|)} \tag{3.4}$$

In both equations, M1 and M2 are messages sent to a measured device.

MDI has better results than SNI according to [17]. Interesting areas are smaller than in SNI, which makes MDI more precise. Unlike SNI, MDI_t is more precise than MDI_{max}.

The third approach is a CEMA (correlation electromagnetic analysis) indicator. CEMA is an improved DEMA 2.2 that takes noise into account and therefore leads to a smaller number of measurements. The CEMA indicator is linked to the number of messages needed to perform a CEMA. It can be thousands of messages, and CEMA needs to be done on each point. This makes it more time-consuming. Conducting this method can take a few days, while MDI or SNI can take only a few minutes.

Results can be calculated in two ways. First one calculates the minimum number of messages that need to be sent to the device for computing the right key using CEMA. The second way sends a fixed number of messages and calculates the ratio of the maximum value of the CEMA peak of the right key. The results of CEMA indicator are similar compared to MDI, but this method takes longer time than MDI.

Chapter 4

Measuring Equipment

In this chapter, we will focus on the equipment we will need in the realisation part of this paper. Schematic 4.1 shows our working environment and all connections between devices.

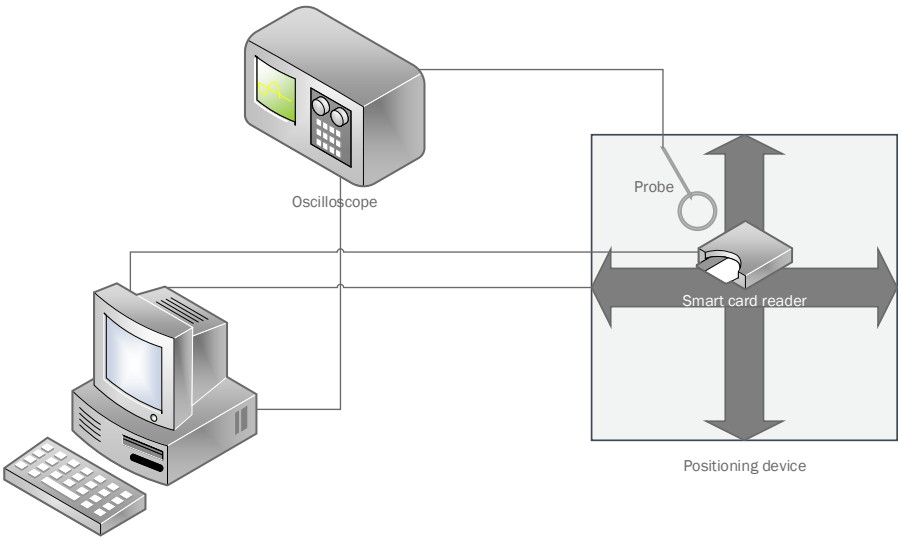


Figure 4.1: Working environment

As we can see in Figure 4.1, we will need an oscilloscope that can communicate and get commands from a computer, a smart card, a card reader connected to the computer, and a positioning device, which will position probes attached to the oscilloscope.

4.1 Oscilloscope

An oscilloscope is an electronic test instrument. It displays the value of an electric signal over time. We can find the amplitude of a signal on the Y-axis and time along the X-axis. It is used for displaying the shape of a waveform, measuring amplitude and frequency of a signal and for detecting glitches and noise in a signal.

Most common types of the oscilloscope are an analogue and a digital oscilloscope. The main difference between these two types is that an analogue oscilloscope shows waveform in the original form, whereas a digital oscilloscope reads a voltage on an input channel, amplifies the signal, conditions the signal and converts the signal with an Analog-to-Digital Converter (ADS) into digital numbers and then stores them in digital format. ADS samples the voltage into an n-bit sample taken every t seconds.

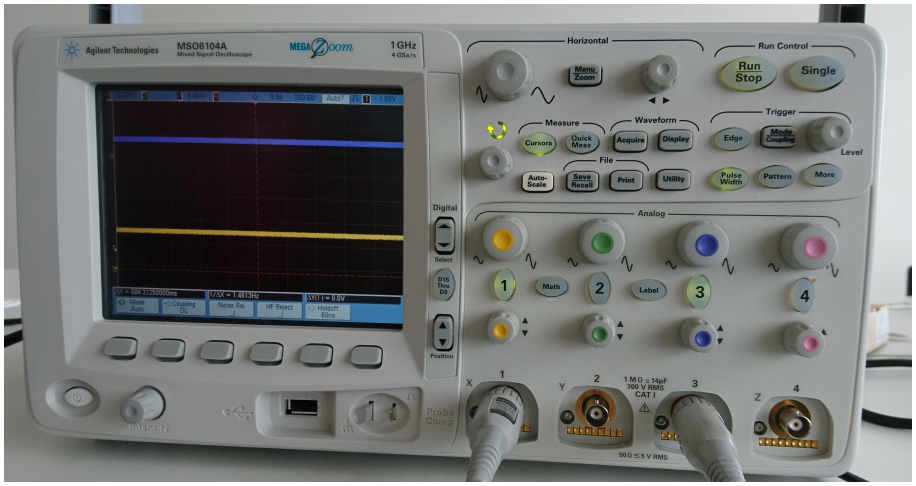


Figure 4.2: Oscilloscope

Essential parts and controls are a display that shows the current status of a signal, graticule, analogue input channels, analogue probes, trigger control etc. 4.2. More information about how oscilloscopes work and what they consist of can be found in [18].

4.1.1 Communication With Oscilloscopes

Most oscilloscopes can communicate with a computer via USB or LAN interface. An oscilloscope usually has a range of SCPI commands (Standard Commands for Programmable Instruments). A computer can send these commands to the oscilloscope, and the oscilloscope will send its response back. Program for controlling oscilloscope can be written e.g. in C/C++ or Visual Basic. A full description of commands and their usage for the used oscilloscope can be found in [19].

4.2 Near-Field Probes

This section describes properties of antennas and probes. More information about antennas and probes and how to find the best one for specific measurements can be found in [11].

Probes are used to measure radiation efficiently. The probe is an essential part of a measurement chain. According to [11], every probe has several fundamental properties, e.g. radiation pattern, polarisation gain, impedance and bandwidth.

- **Sensitivity**
  Some probes are more sensitive towards electric or magnetic fields. Some can be a combination of both.

- **Receiving/Radiation Pattern**
  Receiving/radiation pattern is a spatial sensitivity of the sensor towards electromagnetic fields. It defines suitability of a sensor for a specific task. It is defined by the type or topology of the probe and corresponding design parameters.

- **Polarisation**
  Polarisation of a wave defines behaviour and orientation of the tip of the total electric field vector in a fixed plane over time. The polarisation of the probe corresponds to the polarisation of the wave transmitted or received by the probe.

- **Gain**
  Gain is defined as a maximum of the ratio of the radiated power per unit of space as a function of (θ, ϕ), where θ and ϕ are two of the three parameters from the spherical coordinate system, to the power fed to the probe per unit of space. Losses in the probe are defined as a difference between gain and directivity. Directivity is defined as a maximum of the ratio between the radiated power per unit of space (θ, ϕ) and the mean radiated power per unit of space.

- **Input Impedance**
  Input impedance influences the efficiency of energy transfer to and from the probe.

- **Bandwidth**
  Bandwidth is defined as a frequency range where the probe satisfies some imposed specifications.

Shielded-Loop

Probes can have a shielded or a non-shielded loop. The difference can be seen in Figure 4.3. The most important difference is a gap in the shield in a shielded loop. The same as in an unshielded circuit, induced voltage at the terminals is generated through the change in magnetic flux captured by the loop. It behaves according to Faraday-Lenz law (Equation 1.1). The voltage is induced over the slit at the outer surface of the conductor. The inner side of the shield is electrically separated from the outer side of the shield. The inner surface of the shield and inner conductor bring together transmission line, which is driven by the induced voltage over the slit. The probe itself is a shield. The current runs on the outer side of the shield. When the current gets to the gap, it will return along the inside of the outer conductor, which induces the current in the inner conductor.

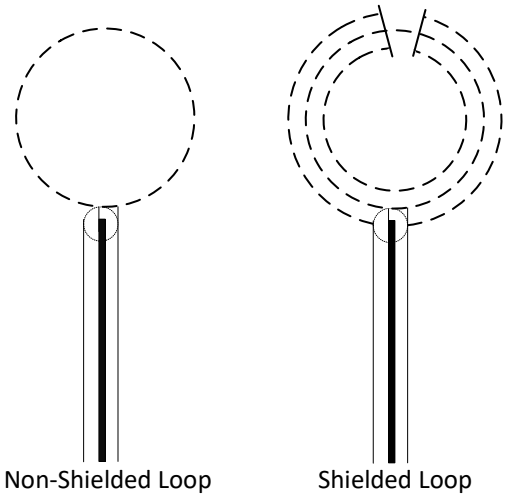


Figure 4.3: Example of non-shielded and shielded probe

4.2.1 RF2 Near-Field Probe Set

In this paper, we will use probes from RF2 Near Field Probe Set [1]. Set consists of four probes, which are all passive near-field probes. They can

measure electric and magnetic fields from 30 MHz to 3 GHz. They all have a current attenuating sheath and therefore are electrically shielded. They can be connected to a spectrum analyser or an oscilloscope with a 50 Ω input.

- **RF-R 400-1** 4.4
  This probe is highly sensitive. It is suitable for measurements up to 10 cm. It has a spatial field structure.

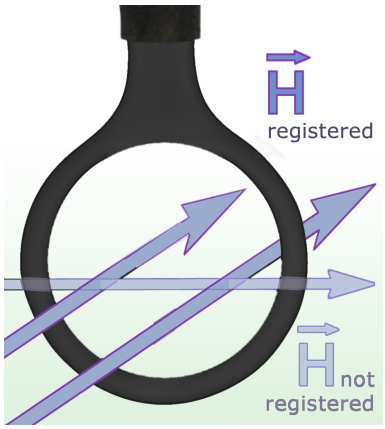


Figure 4.4: RF-R 400-1 measuring principles [1]

- **RF-R 50-1** 4.5
  This probe covers less of the magnetic field due to its smaller diameter, which is 10 mm. Therefore, it is less sensitive than RF-R 400-1, but it has a higher resolution than RF-R 400-1. It is suitable for measurements up to 3 cm.

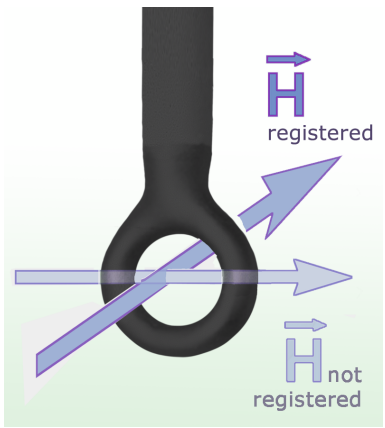


Figure 4.5: RF-R 50-1 measuring principles [1]

- **RF-U 5-2** 4.6
  This probe is suitable for detecting magnetic fields at broad conducting paths, cables, etc. The probe head can be put directly on the component.

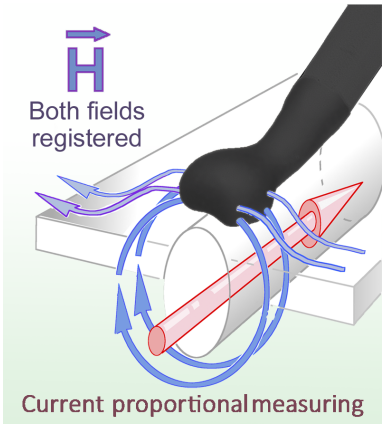


Figure 4.6: RF-U 5-2 measuring principles [1]

- **RF-B 3-2** 4.7
  The measurement coil of this probe is at 90° angle to the probe shaft. It touches the surface of the printed circuit board directly. In contrast to RF-U 5-2 probe, magnetic lines that enter the probe laterally are not detected.

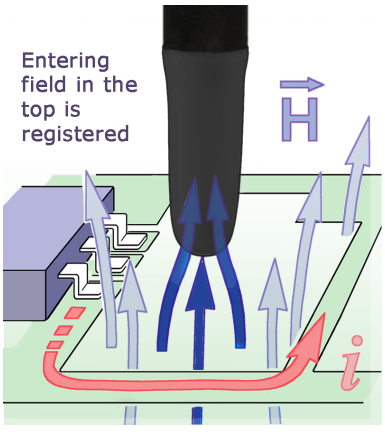


Figure 4.7: RF-B 3-2 measuring principles [1]

4.3 Positioning Device – 3D Printer

A technology used for 3D printing belongs to the category of CNC machines. CNC machines are machines controlled by a numerical computer. There is a

wide range of these machines. They can be used in agriculture, in manufacturing and many other fields. An example of a CNC machine could be a milling machine, a plasma cutter or a drill. These machines have preprogrammed sequences of machine control commands and communicate only by listening to these commands.

In this thesis, we will use 3D printing technology for positioning probes in three-dimensional space. We will use the RepRap technology. It is currently very popular type of an CNC machine. RepRap is a form of a free 3D printer. It is a self-replicating machine and the first of the low-cost 3D printers. We will work with a model called RebeliX, beacause it is easily accessible and it was available for us. This model is inspired by Rebel II and Prusa i3. It is an open-source project and a part of the RepRap project. The complete manual on how to construct this 3D printer can be found in [2].

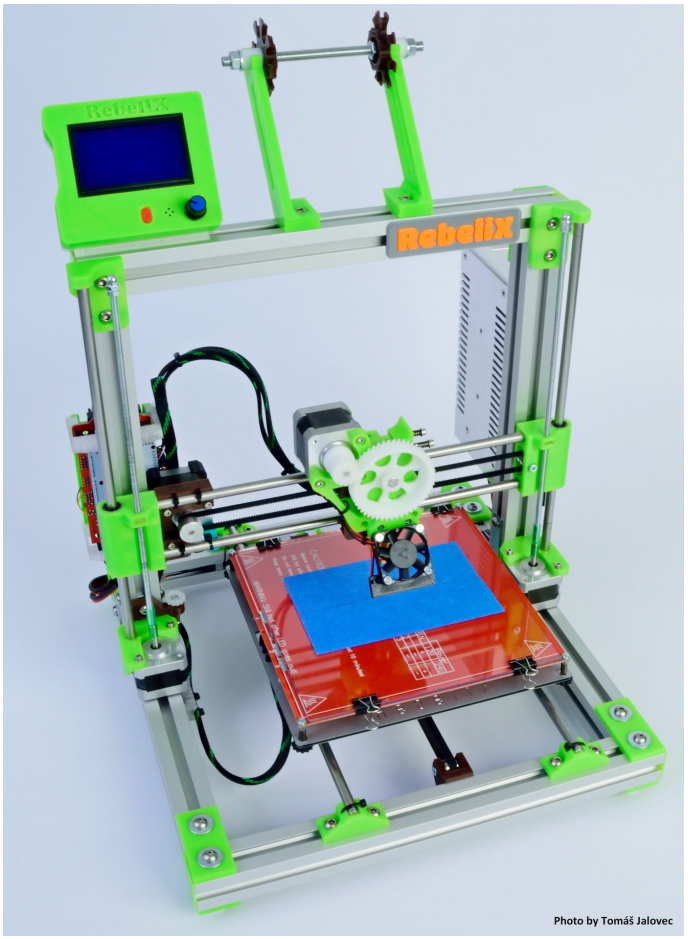


Figure 4.8: RebeliX 3D printer [2]

4.4 Smart Card

A smart card is a plastic card that is used as a storage of information. It is an authentication factor. Examples of smart cards include a payment card, an electronic ticket, an entry card etc. Smart cards are defined in a standard called ISO 7816. ISO 7816 does not determine the electronics, and a smart card can have just a memory chip, or more likely nowadays, a microcontroller and a memory chip.

The microcontroller of the smart card is either configured using a manufacturer-supplied API of commands, or it can be programmed with a custom program during manufacturing or in the field. They do not contain any battery; smart card reader or writer supplies the power. Security measures include symmetric authentication and encryption or asymmetric schemes.

We can divide cards into groups by API they use for communication. The first group is contact smart cards. It means that we must insert the smart card into a smart card reader or writer with a direct connection to a conductive contact plate on the surface of the card.

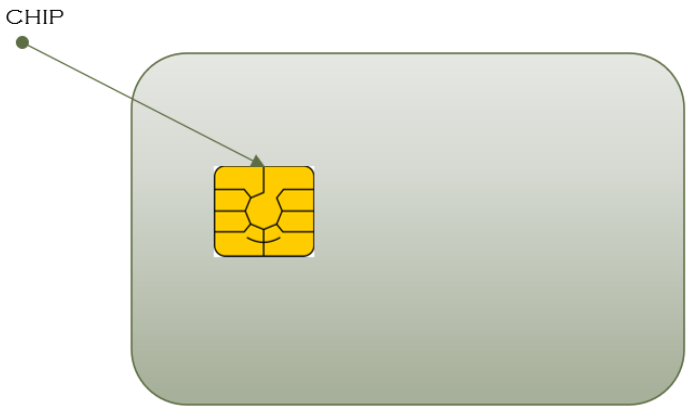


Figure 4.9: Contact smart card

They can contain up to eight contact PINs, see Figure 4.10.

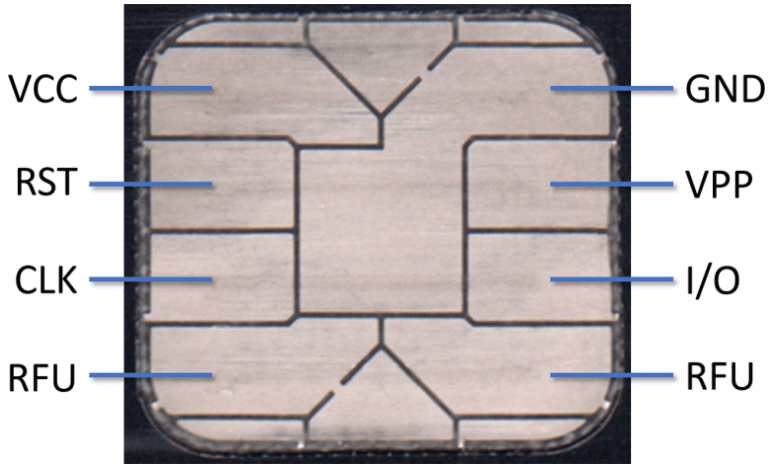


Figure 4.10: Contact smart card PINs [3]

The second group is contactless smart cards. Contactless smart cards communicate using a radio frequency signal, with a typical range of less than 2 feet. Communication of contactless smart cards is based on technology similar to Radio Frequency ID tags (RFID) used in stores to counter theft and track inventory.

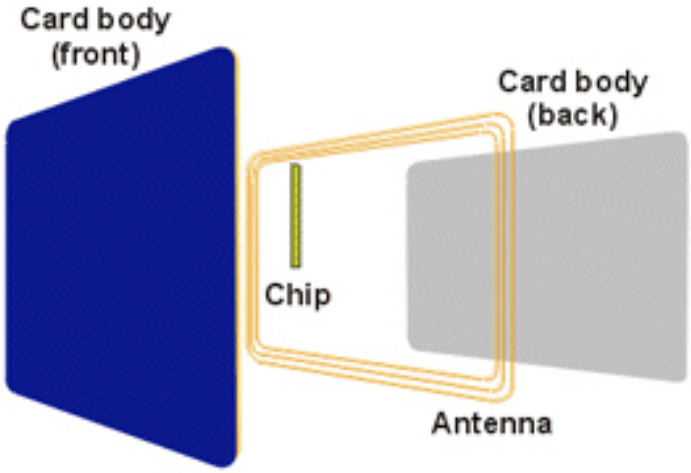


Figure 4.11: Contactless smart card [4]

The third group is a combination of the two previous groups. The first one is a hybrid smart card which contains more than one chip inside. The chips have no interface with each other. The second one is a dual interface smart card. It is a combination of contact and contactless smart card using a

single chip with both interfaces. It is mostly used in banking applications as a payment card.

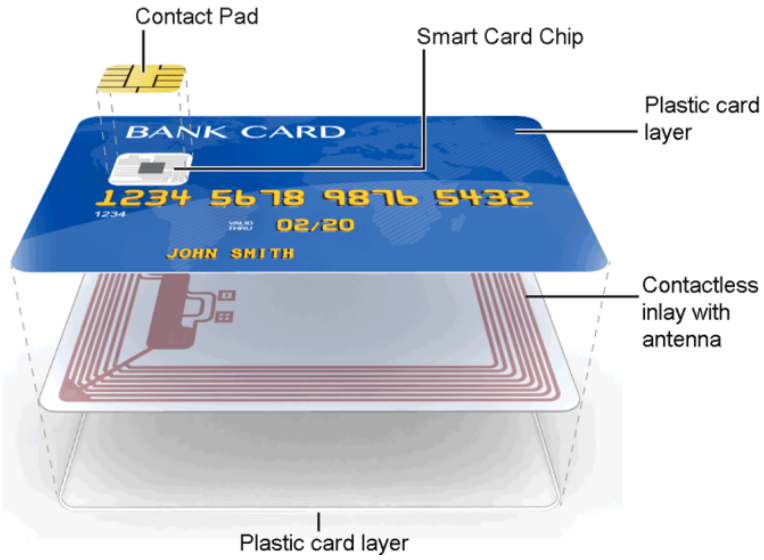


Figure 4.12: Dual smart card [3]

Cards can also be electrical, optical or based on the electromagnetic signal [20].

In this thesis, we will work with an AVR card. This card contains ATMega163 and 256K EEPROM, see Figure 4.13. It can be programmed in C or Assembler. It is a contact card, nowadays mostly used for educational purposes. It has no protection against side channel attacks.

4.4.1 Smart Card Programming

In this paper, we will use AVR Dragon programmer (Figure 4.14) for programming smart cards. We can write a program e.g. in C and use AVR Studio for uploading the program into the smart card.

Figure 4.13: ATMega Card

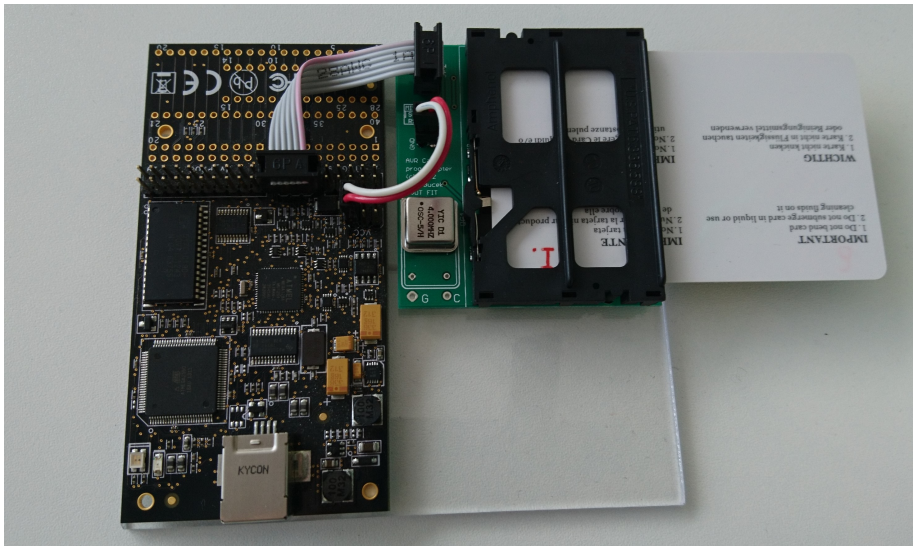


Figure 4.14: AVR Dragon Programmer with custom smart card interface adapter

4.4.2 Smart Card Reader

For communication with the smart card, we used ACR38U-I1 smartcard reader from the company Advanced Card Systems Ltd. [21]. It supports smart cards and microprocessor cards with protocols T=0 and T=1. This card reader is considered to cause noise in electromagnetic field, so the measurements can be influenced and distorted by using this reader. We chose to use this model because it was readily available for our use.

Chapter 5

Realisation of the Working Environment

We can see the whole working environment in Figure 5.1. The working environment consists of a positioning device, a smart card and a smart card reader, a probe, a holder of the probe, an amplifier, an oscilloscope and a computer. This chapter describes the realisation of the working environment, rebuilding a 3D printer into a positioning device, designing and printing holders for the probes.

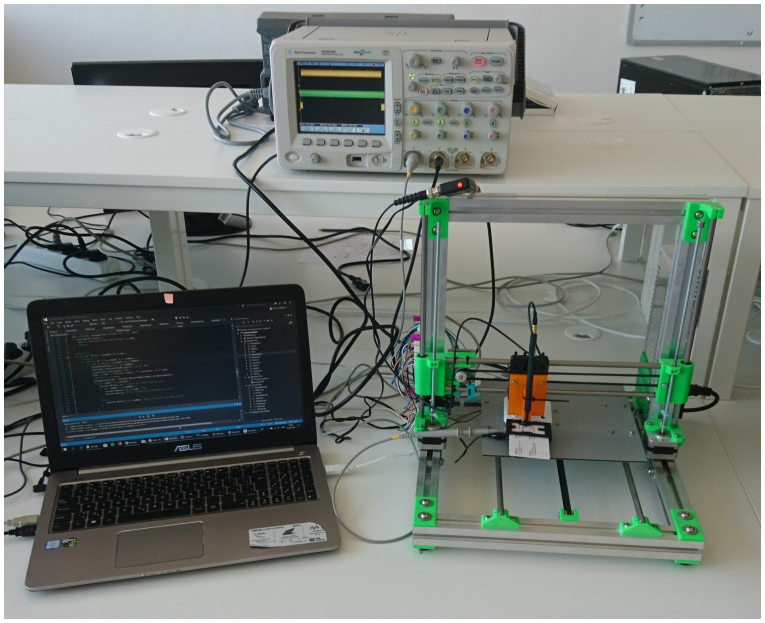


Figure 5.1: Working environment

5.1 Positioning Device

We used 3D printing technology, specifically 3D printer RebeliX because we have this technology at the faculty. It is a low-cost printer and it is easily accessible. The printer frame was already built, we needed to connect stepper motors and set the printer's firmware.

We used Minitronics board, which claims to be a natural solution that fits 90% of the 3D printers. It has Atmega1281 processor with 128 KB memory, running at 16 MHz. It can be connected to a computer via USB cable. The board is compatible with Arduino. The schematic of the board can be seen in Figure 5.2. The Minitronics in the picture is version 1.0; we used version 1.1. There are only slight changes between these two versions. More information about Minitronics, its properties and technical specification can be found in Minitronics v1.1 Datasheet [22].

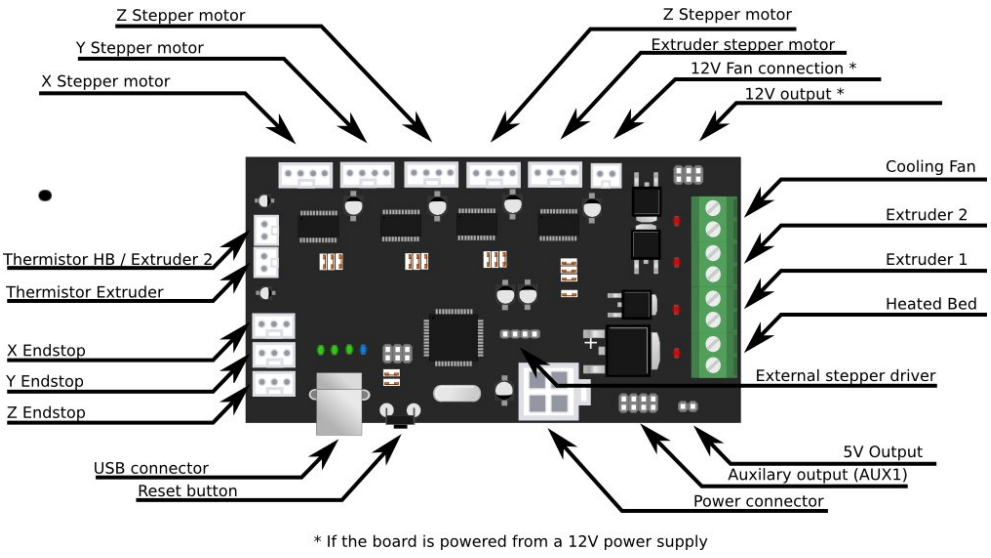


Figure 5.2: Schematic of Minitronics 1.0 [5]

We connected X, Y and Z stepper motors and endstops. We did not connect extruder stepper motor or extruders, because we did not need them; we have a measuring probe instead of an extruder. We also did not have to connect a cooling fan, because we used DRV8825 stepper motor driver [23] that does not need a cooling system for the used electric voltage (12 V). We needed to connect a thermistor extruder. The board has three analogue inputs for temperature (the schematic shows only two of them), one of which is used for temperature control. However, we do not use an extruder; we decided to connect a fixed resistor instead of the thermistor. The resistance of the

resistor is 100 kΩ, which is the resistance of a commonly used thermistor type at room temperature. It makes the printer usable even without an extruder and a heated bed. After that, we connected the board to a power source and a computer.

As a firmware we used Marlin-RC, which is an open-source 3D printer firmware. We had to make a few changes in the firmware's configuration because we did not use a printhead. We made changes in the file `Configuration.h`. We needed to double the default steps per unit because we used different stepper motor driver. The default value was set to 16 micro steps, but the motor stepstick DRV8825 support 32 micro steps. We also changed `X_MAX_POSITION` from 200 to 190 in order to avoid crashing the X axis into a hard mechanical limit. The length of the X axe should be 20 cm, but the length in this printer is a little bit less.

After changing the firmware, we were able to program the Minitronics board with Arduino using Marlin firmware.

5.2 Holders for Probes

To be able to mount probes to the positioning device, we needed to design and print holders for the probes. We designed the holders in OpenSCAD, which is an open source program for creating 3D objects. After that, we printed them on another 3D printer that was configured for normal usage.

We needed to design and print two different holders according to the shapes of probes (subsection 4.2.1) used for measurements. One holds a probe in a horizontal position and the other in a vertical position. Figure 5.3 and Figure 5.4 show the design of the holders.

Holders can be modified and printed according to dimensions of the chosen 3D printer. Size of the holes for screws is parametrised, as well as the distance between the holes.

After we finished the design, we exported the holders into STL format. We used Slic3r to slice the model and generate commands for the printer. Then we printed the holders using Pronterface on the 3D printer. We can see the printed holders in Figure 5.5 and Figure 5.6.

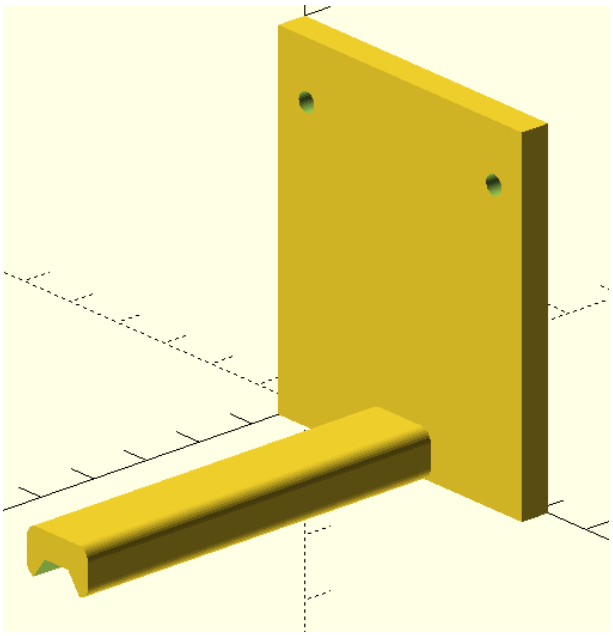


Figure 5.3: Design of horizontal holder for probes 4.4, 4.5

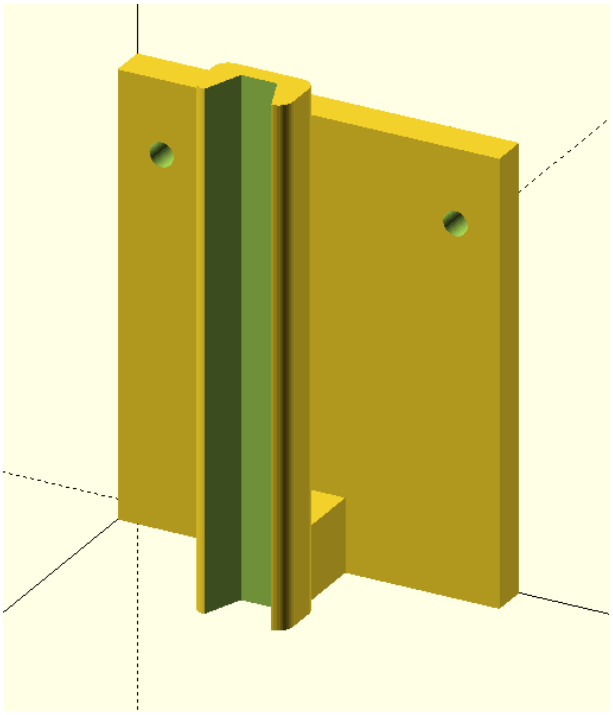


Figure 5.4: Desing of vertical holder for probes 4.7

Figure 5.5: Printed vertical holder for probes 4.7

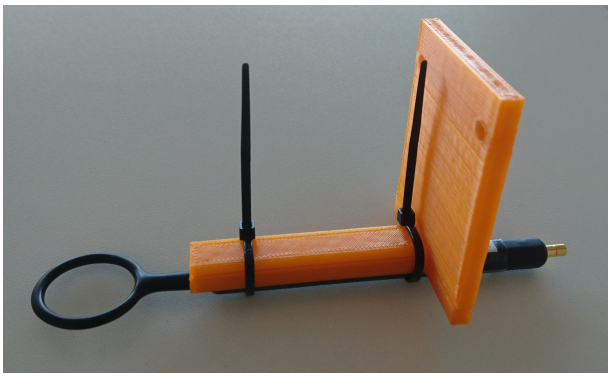


Figure 5.6: Printed horizontal holder for probes 4.4, 4.5

Chapter 6

Implementation of the attack

We will conduct an attack against a smart card in different positions with respect to its chip. We need to communicate with the positioning device to change locations. Then we need to send commands to an oscilloscope when it should measure data, get measured data from the oscilloscope and save them for future usage. We have to send a plaintext to the smart card and save the received ciphertext using a smart card reader.

In each position, we will try to guess the key used by the smart card. We will use differential electromagnetic analysis to do so. Finally, according to the measured data and the success of DEMA, we will visualise the results in graphs using Wolfram Mathematica.

We will implement the program in Visual Studio in C++. The program will be running only on Windows due to usage of Windows-specific libraries (Keysight IO Libraries Suite [24]). We can see the whole structure of the program in schematic Figure 6.1.

6.1 Communication

As we wrote in the previous paragraph, the computer needs to communicate with an oscilloscope, a smart card and a positioning device. Some parts of the communication were already implemented and only slightly changed, some of them were written from scratch. Communication with an oscilloscope and a smart card was written by Petr Vyleta for use in laboratory exercises of the Hardware Security course (MI-HWB).

6.1.1 Oscilloscope

Communication with oscilloscope was already written. Even though it was written for a different oscilloscope (for Agilent DSO-X 3012A [19], we used

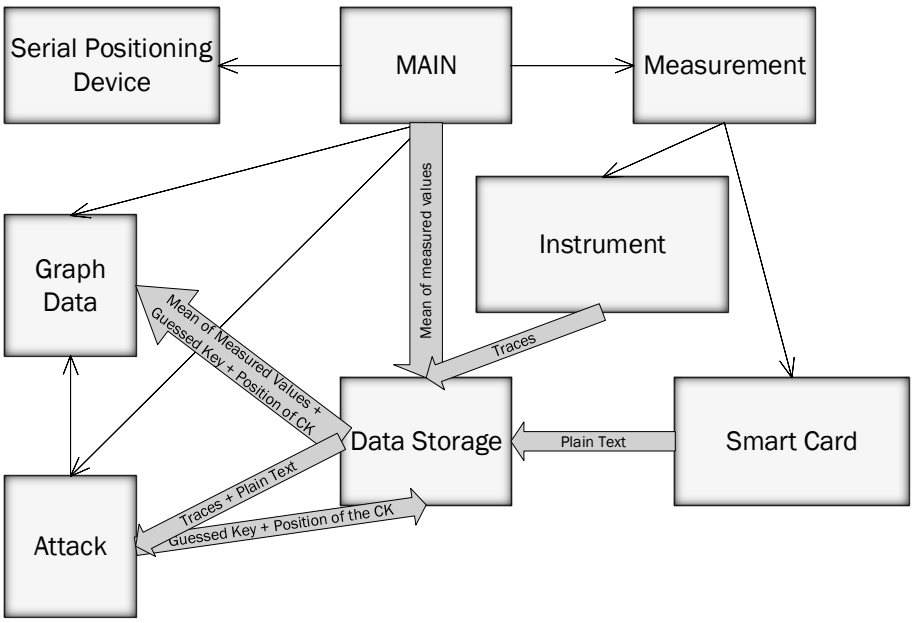


Figure 6.1: Scheme of the implemented program

Agilent DSO 6104A [25]), the main methods for sending commands and saving measured data were functional. The class consists of methods for sending commands, receiving messages and connecting to an oscilloscope. The only methods that did not work with the used oscilloscope were methods for saving and loading configuration of the oscilloscope.

The program uses VISA (Virtual Instrument Software Architecture) for communicating with the oscilloscope. To use this program, you need to install VISA and add it to the library path in Visual Studio.

6.1.2 Smart Card

Communication with a smart card was already written as well. It consists of methods that connect a computer to the card, send plaintext to the card and receive ciphertext from the card using APDU (Application Protocol Data Unit).

The program uses PC/PS interface to communicate with the smart card. Specifically, it uses WinSCard API. It communicates using protocol T=1, which means it communicates in blocks.

6.1.3 Positioning Device

The communication with the positioning device had to be implemented by us. As we wrote in the previous chapters, we used 3D printer technology as a positioning device, therefore the communication with the positioning device is a communication with a 3D printer. The controlling software sends G-Code commands (see Figure 6.2) to the printer to let the printer know where to move. G-Code is a language used with CNC devices.

```
1   G17 G20 G90 G94 G54
2   G0 Z0.25
3   X-0.5 Y0.
4   Z0.1
5   G01 Z0. F5.
6   G02 X0. Y0.5 I0.5 J0. F2.5
7   X0.5 Y0. I0. J-0.5
8   X0. Y-0.5 I-0.5 J0.
9   X-0.5 Y0. I0. J0.5
10  G01 Z0.1 F5.
11  G00 X0. Y0. Z0.25
```

Figure 6.2: G-Code example

The printer uses serial communication for sending and receiving data. We used class for serial communication written by Thierry Schneider.

We implemented a method for connecting a computer to the printer. In the connection method, it is also necessary to do homing of the printer to know future positions after moving and for the printer to know the boundaries where to stop. Then we implemented two methods for positioning the probe. Both methods need G-Code with specified positions where to move.

6.2 Positioning

Probes can be positioned and moved in three dimensions. In the program, we can choose dimensions of the space which will be examined. The length can be different along each axis. We can also set the size of the gap between each measurement in a configuration file. We can see the diagram of the measured area in Figure 6.3.

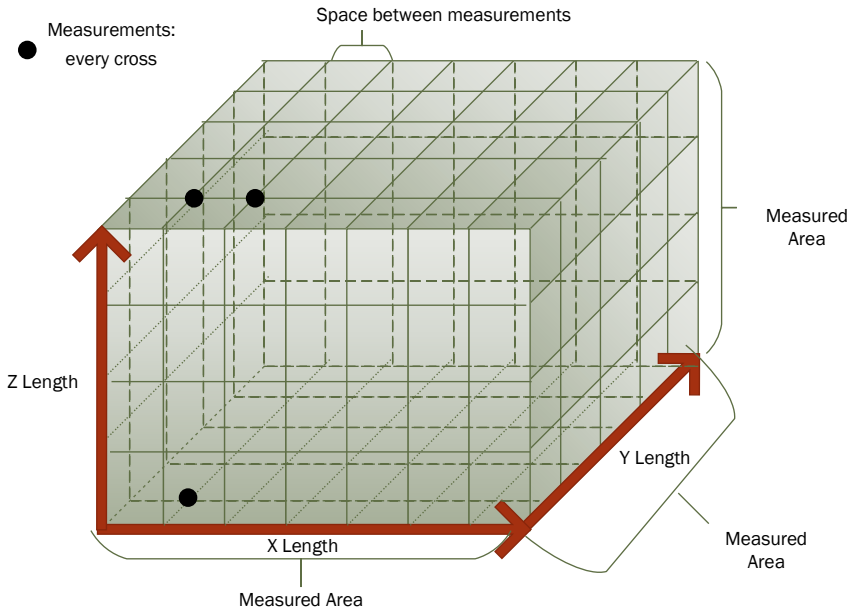


Figure 6.3: Scheme of measured/attacked area

6.3 DEMA Attack

We implemented differential electromagnetic attack in C++. The program attacks bytes one by one. The program uses threads, one thread for each byte; the number of the threads running at the same time can be specified in a configuration file.

Before we attack the bytes, there is an initial part. DEMA takes initial information such as the plaintext, the measured traces and the length of each measured trace from files. The directory where the data is stored is also specified in the configuration. In this part, data length which we will use in DEMA (part of the measured data) and data start position are loaded from the configuration file. These two parameters specify the part of the measured data that we use for the attack. After that, plaintext and traces are loaded from files.

After initial part, we attack the key bytes. We use output of function *AddRoundKey()* as a place to attack. We use Hamming weight to model the hypothetical values of electromagnetic emission. Finally, to get the right key, we use the correlation of matrices with Hamming weight and measured traces. The whole diagram describing the attack on one byte of the key can be seen in Figure 6.4.

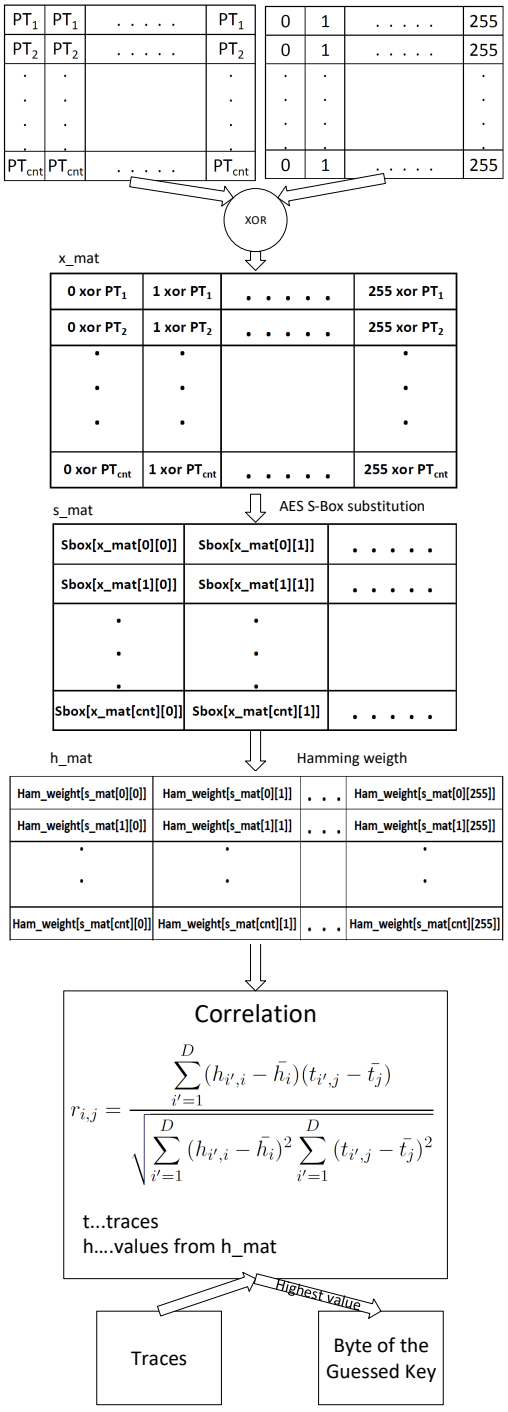


Figure 6.4: Schematic of an attack on one byte

6.3.1 Time Requirement

DEMA attack can take quite a long time, depending on the number of measurements. The first prototype of this program was very slow; it used STL containers, such as vector. After rewriting vectors to arrays, the program got considerably faster. We tried to use libraries for faster computation with matrices such as GSL, but the program got even slower. The next thing that sped the program up was to change correlation calculation, which is the most time-consuming part of the program. Instead of iterating one by one and computing correlation for one value in one iteration, the program now calculates the correlation for eight values in one iteration. This change sped up the program about twice. Finally, we decided to use threads to make the program faster. We use one thread for calculating one byte of the key. DEMA is a computationally challenging problem, and it is not possible for every computer to compute all sixteen bytes at once. Therefore, this parameter is configurable. The number of threads running at the same time can be set in the configuration file.

6.4 Configuration File

Some parameters can be set in a configuration file.

- `READER_NUMBER` - the number of the smart card connected to the computer,
- `NUMBER_OF_MEASUREMENTS` - the number of measurements for each position,
- `SPACES` - the length of the space between measurements on axis X and Y,
- `Z_SPACES` - the length of the space between measurements on the Z axis,
- `OUTPUT_FOLDER` - path to the output folder for measured data = input folder for DEMA and graphs,
- `PRINTER_PORT`,
- `MEASURED_DATA_LENGTH` - data length used for DEMA,
- `DATA_START_POSITION` - starting position of the data for calculating DEMA,
- `NUMBER_OF_THREADS` - the number of threads running at the same time in DEMA calculation,
- `CIPHER_KEY` - used cipher key for AES encryption.

6.5 Graphs

As we wrote in the previous section, most of the graphs were created using Wolfram Mathematica. There are several ways how to establish the best place to attack, as is described in chapter 3. We visualised graphs and the best spots according to several factors.

The first approach was to compute absolute mean value of measurements at each position, find the highest one and consider that to be the best spot to attack. The second one was to get the best spot by counting bytes of the key that the program guessed correctly. In the third approach, we calculated the position of the correct key in the correlation matrix and visualised data according to this factor. For the second and the third approach, it is necessary to know the right cipher key used by the smart card.

Chapter 7

Testing

This chapter describes how we tested the implemented program and presents the results that we obtained during testing.

7.1 DEMA

First of all, we needed to program the smart card to encrypt using AES. We used a program written by Matthias Bruestle using files from the Chair for Embedded Security (EMSEC), Ruhr-University Bochum. AES implementation in this program was written by Jeff Tikkanen. The program was modified by Filip Stepanek for smart card education at FIT-CTU. The program is written in C, with small parts written in assembler. We programmed the smart card using AVR Studio 4. During the programming, we could also choose our key to know if the DEMA was successful.

In order to have better results when calculating DEMA, we need to know which data part of the encryption we need to save. We are attacking the initial part of the cipher; we need to cut the signal according to the part of the encryption we want to attack. In this implementation, we dropped the first 16000 samples of the traces, took the next 50000 samples and calculated DEMA from these. The counts are parameters and can be set differently in the configuration file. To know when the encryption starts, we need to detect the trigger signal. In order to detect it, we used measuring adapter in Figure 7.1. This adapter allows us to detect when the encryption starts and save traces accordingly. We put the measuring adapter into the smart card reader and the smart card into the measuring adapter.

There had to be some modifications done to the card reader to get a stronger signal. Since the reader had a plastic component over the chip, the radiation was lower and we could not place the probe close enough to the chip.

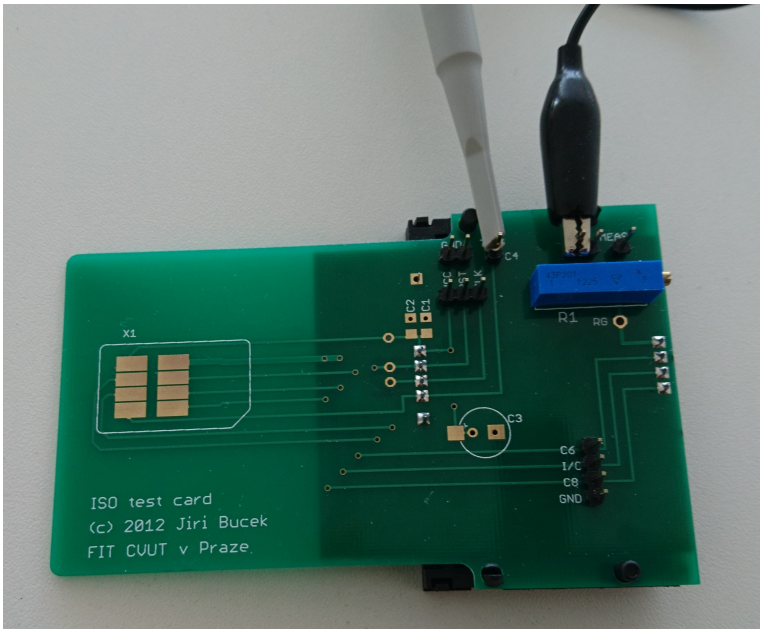


Figure 7.1: Measuring adapter

We decided to cut a part of the plastic component out to get the probe closer to the chip of the smart card, see Figure 7.2.

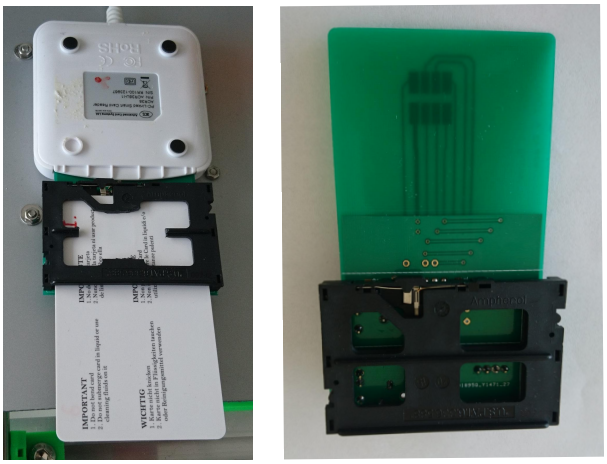


Figure 7.2: Measuring adapter with plastic part and with the part cut out

Initially, we were calculating DEMA after one set of measurements without moving the probe. First of all, we found the position to do the measurements at by measuring the area over the chip of the smart card and placing the probe above the place where the radiation was highest. We calculated DEMA for a

different number of measurements.

After that, we started measuring the signal and calculating DEMA in different positions. It turned out that the best spot for guessing the cipher key correctly is not the spot with the highest signal. We calculated DEMA again using a different number of measurements in the place where we were able to guess the highest number of the key bytes correctly. Results presented in this chapter are the results from the spot with the highest number of the correctly guessed bytes of the key, not from the place with the highest signal.

We tested the program for 50, 200, 300, 400, 600 and 800 measurements. We did each number of measurements for ten times except for 50 measurements. Figure 7.3 shows the number of correctly guessed bytes of the cipher key. For 200 measurements, the best run guessed 10 bytes of the key correctly; the worst one guessed only 2 bytes correctly. For 300 measurements, always at least half of the bytes was guessed correctly, but none of the runs guessed the whole key correctly. The first time the entire cipher key was guessed correctly was for 400 measurements. 600 and 800 measurements had 100% success guessing the key.

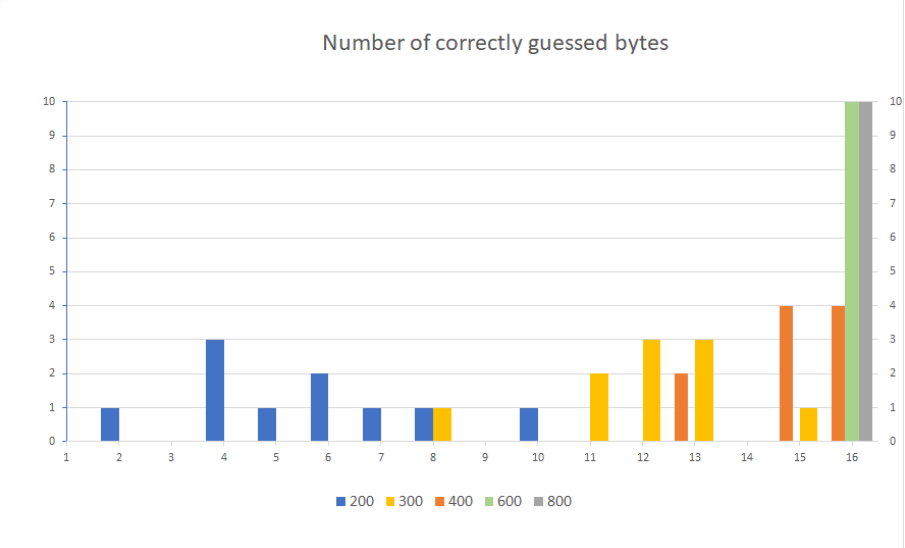


Figure 7.3: Number of correctly guessed bytes of the cipher key

Figure 7.4 shows success in guessing each byte. It shows how many times in ten measurements was the exact byte of the cipher key guessed correctly. We can see that for 200 measurements some of the byes were never guessed correctly, while some of them were guessed for 7 or 8 times. However, there cannot be seen much relation between the position of the byte in the cipher key and number of correct guesses. For example, byte number 14 was never

guessed correctly for 200 measurements, but it was always guessed correctly for the rest of the measurements, whereas byte number 1 was guessed the same amount of times for 200 and 300 measurements.

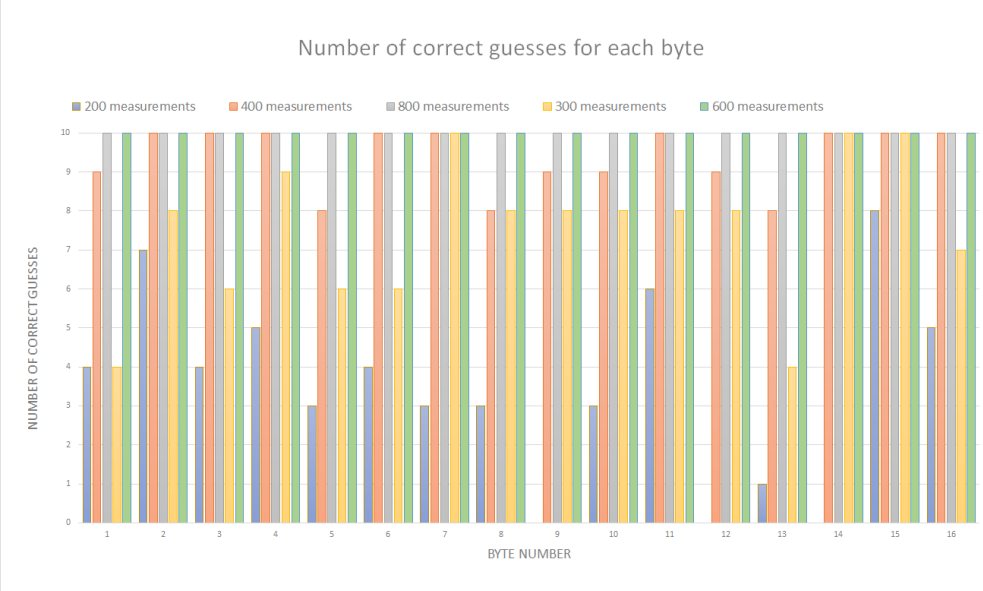
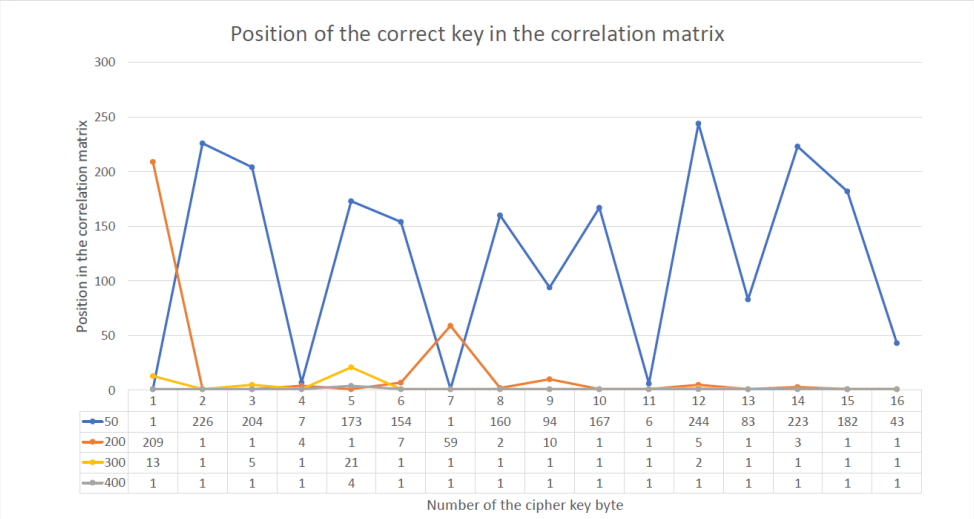


Figure 7.4: Correctly guessed byte of the cipher key

The third graph 7.5 shows the position of the correct cipher key in the correlation matrix. It shows the first measurement for each number of measurements. Tables for all ten measurements of each number of measurement can be found in Appendix B. We can see that for 200 measurements the position can be high and the correct key can be almost the last one. For 300 measurements, all the positions can be found in the first hundred of the highest calculated correlations. For 400 measurements the positions are in the top ten highest calculated correlations. For 600 and 800 measurements the cipher key was always guessed correctly, and the position is always the first.

Figure 7.6 shows the position of the correct cipher key in the correlation matrix for each byte. It is obvious that for 50 measurements the position of the correct cipher key is mostly in the second half of the correlation matrix, for 200 measurements it is mostly in the first half, and for 300 and more measurements it is almost always in the first position of the correlation matrix.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
50	1	226	204	7	173	154	1	160	94	167	6	244	83	223	182	43
200	209	1	1	4	1	7	59	2	10	1	1	5	1	3	1	1
300	13	1	5	1	21	1	1	1	1	1	1	2	1	1	1	1
400	1	1	1	1	4	1	1	1	1	1	1	1	1	1	1	1

Figure 7.5: Position of the correct cipher key in the correlation matrix for each individual measurement

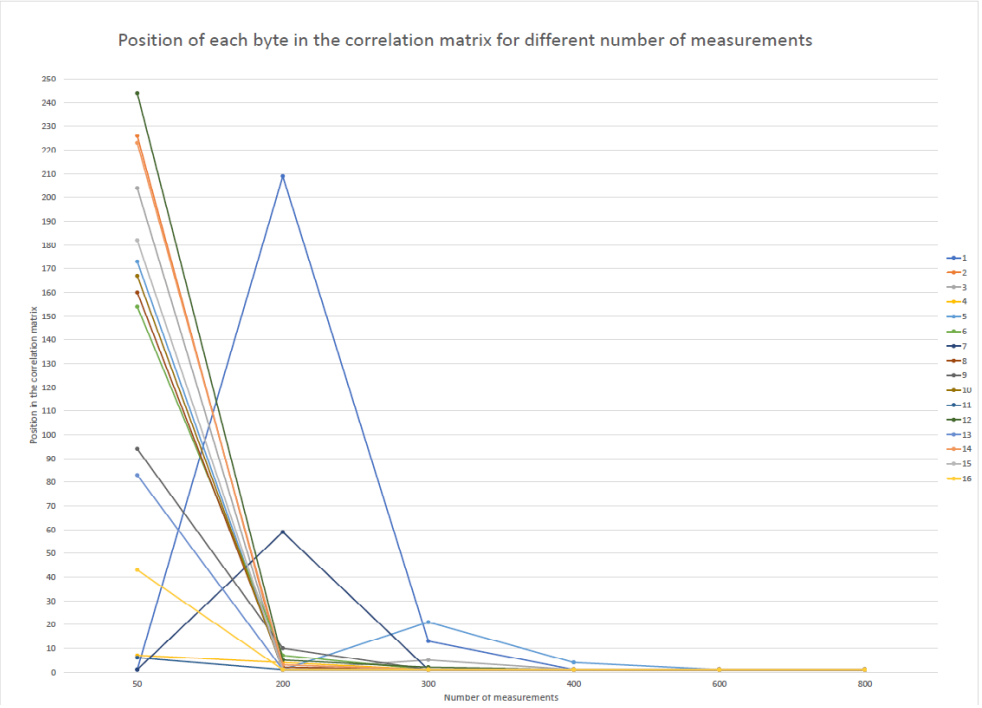


Figure 7.6: Position of the correct cipher key in the correlation matrix for each byte of the key

7.2 2D Positioning

Firstly, we focused on measuring electromagnetic emissions in different positions and calculating the absolute mean value of the emissions. Secondly, we calculated DEMA in each position and counted the number of correctly calculated bytes and positions of the correct cipher key bytes in the correlation matrices.

We conducted measurements over the smart card's chip. Numbers at the graphs labels are millimetres from the homing position of the positioning device. They depend on the location of the smart card within the positioning device. We had the smart card at one place for all of the measurements. These numbers are the borders of the chip on the smart card. We did most of the measurements at positions 43-51 at the X-axis and 35-45 at the Y-axis, which are the dimensions of the smart card's chip. The coordinates mapping to the smart card's chip can be seen in Figure 7.7. We did not cover the whole chip; however, we covered a small area near the chip, because the signal was stronger on one side than on the other.

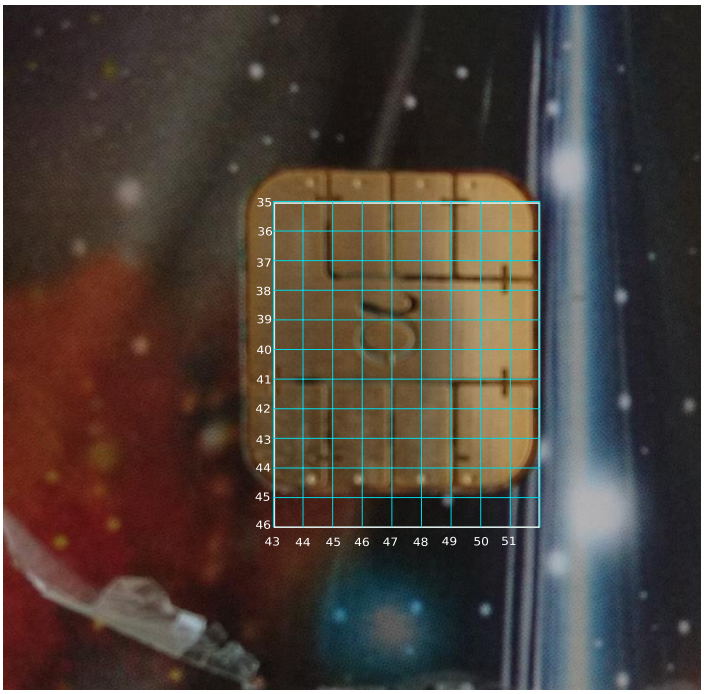


Figure 7.7: Mapping coordinates to smart card's chip (coordinates is millimeters

Firstly, we moved with the probe only in two-dimensional space. We can see different mean values of the emissions in Figure 7.8. This graph shows the intensity of the signal over the chip of the smart card. As we can see, there are two main spots with the highest signal around the centre of the chip. We took the measurements in each millimetre of the chip. The graph covers an area of 12 mm × 13 mm. This measurement was done in a larger space and with only one-millimetre gaps, because we wanted to see how the intensity of the signal varies. It is evident from the graph that the intensity at the borders of the chip is much lower than the radiation intensity in the middle of the chip.

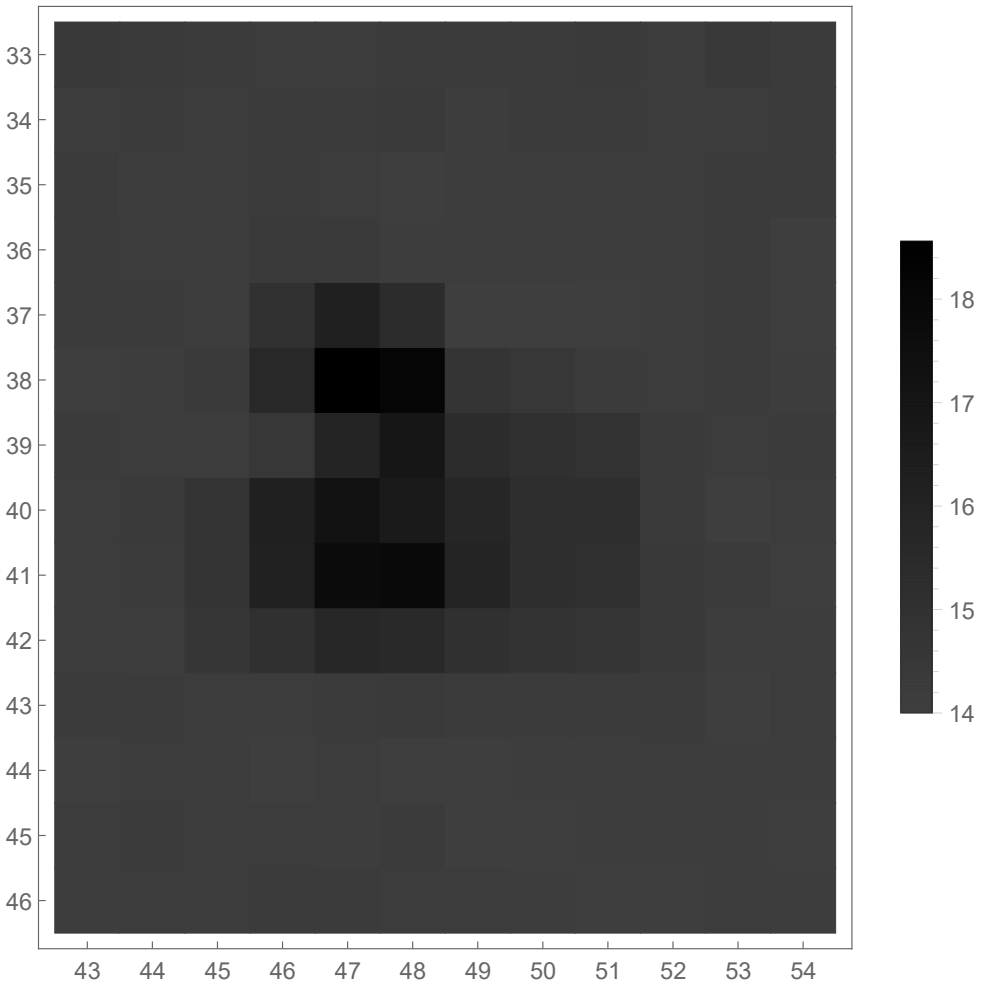


Figure 7.8: Graph of signal intensitity in 2D area

After that, we did measurements again with calculating DEMA and knowing the correct cipher key. This time, we did measurements every two mil-

limetres, computed DEMA in each position, calculated how many bytes of the key were guessed correctly and estimated the positions of the bytes of the correct cipher key in the correlation matrices. We tried different number of measurements; we started with 200, then 400 and 800.

As we can see in Figure 7.9, we barely guessed any bytes of the key for 200 measurements. We can see some correlation between the graph of the mean values in Figure 7.8 and the number of correctly calculated bytes in Figure 7.9. However, we guessed the highest number of the bytes correctly in the position [49,41] and the second highest at the position [49,39], while the strongest signal was at the positions [46-47,41] and [46-47,38]. The strongest radiation did not mean the highest number of the correctly guessed bytes.

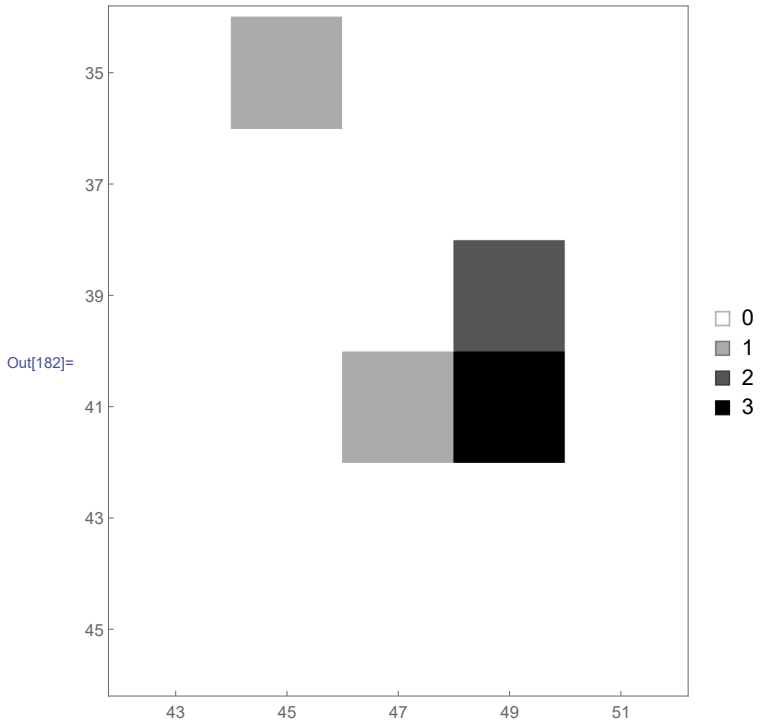


Figure 7.9: Number of correctly calculated bytes of the correct cipher key, 200 measurements

For 400 measurements, the results are better as can be seen in Figure 7.10. In spite of that, there is still no place where it is possible to guess the whole key correctly. The highest number of the correctly guessed bytes of the key is ten. It is in the same position as the highest number for 200 measurements. The second highest number of the correctly guessed bytes corresponds to the position in the graph for 200 measurements 7.9. As previously observed, this measurement also confirms that the strongest signal does not mean the highest

number of correctly guessed bytes of the cipher key.

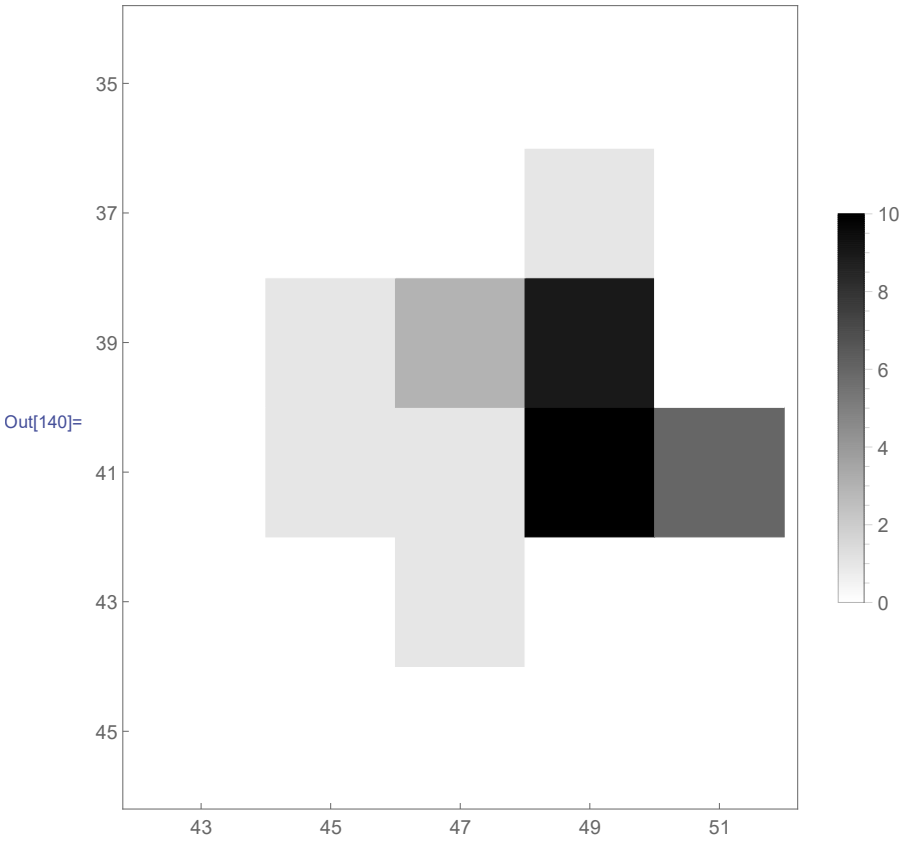


Figure 7.10: Number of correctly calculated bytes of the correct cipher key, 400 measurements

Then we did 800 measurements in each position. We managed to find three places where it was possible to guess the whole key correctly. The number of bytes guessed correctly can be seen in Figure 7.11. The most successful positions correspond with the most successful positions for 200 and 400 measurements. Here we can see that the best positions are in the middle of the Y-axis and on the right side of the X-axis.

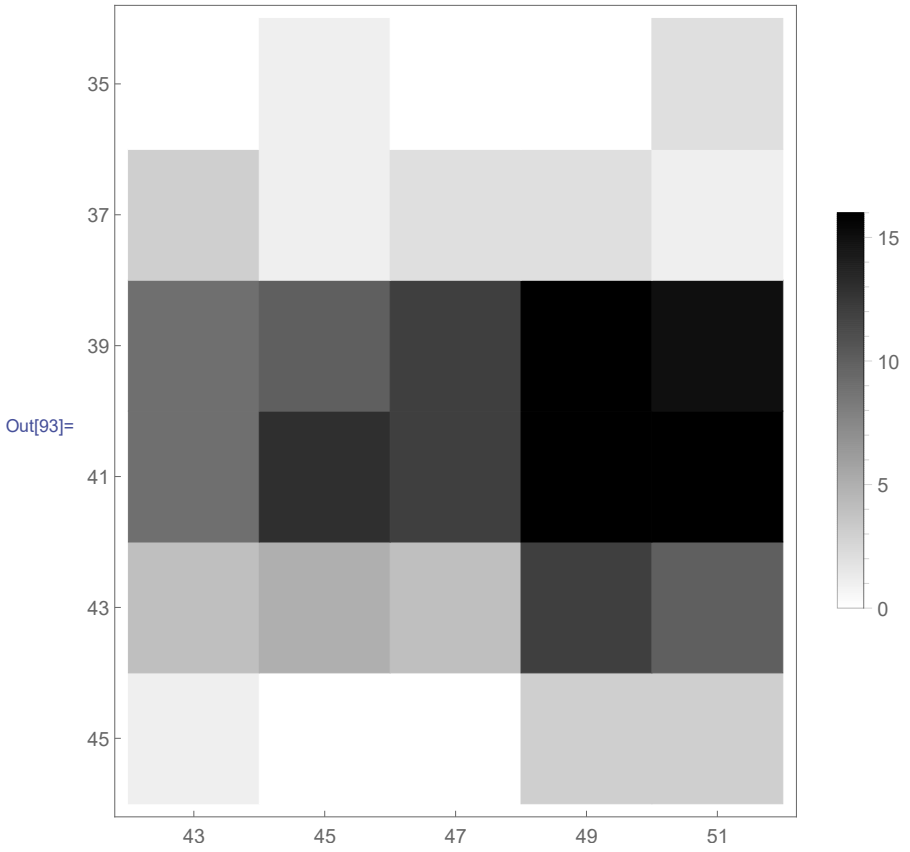


Figure 7.11: Number of correctly calculated bytes of the correct cipher key, 800 measurements

All of the measurements show that the best spot to attack does not have to be the spot with the strongest signal. It means that graphs based only on the mean of the signal measured at that area are not necessarily helpful while searching for the best spot to attack. In all of the measurements, the signal at the best attacking position was average or even below average.

We can see in Figure 7.9, Figure 7.10 and Figure 7.11 that the best spot to attack is the position at the coordinates [41,49]. We can see the place at the chip when we map it to the smart card in Figure 7.12

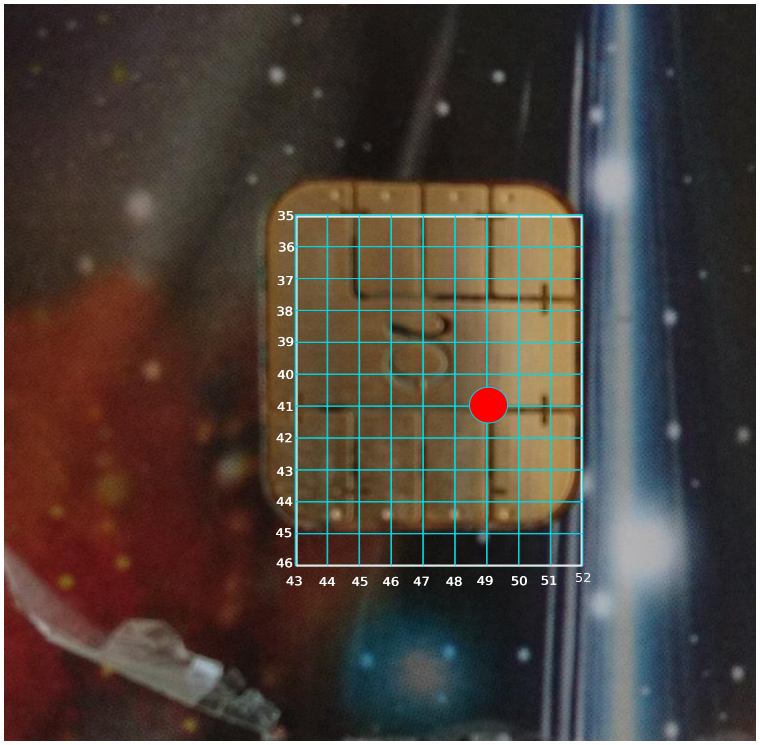


Figure 7.12: Coordinates of the best spot to attack

When we compare success in guessing the key bytes in two-dimensional space to the success in calculating DEMA in one position, it is obvious that the results are better when we calculate DEMA without changing positions. The reason is that the range of the data the oscilloscope sends to the computer is the same range that we can see at the oscilloscope's display. We did not implement adjusting the amplitude of the signal to the intensity of the signal, because we wanted to calculate mean of the signal intensity and the mean diverse from different display adjustment. Therefore the results of the DEMA are not as good as in the first part of testing.

Finally, we counted position of the correct cipher key in the correlation matrix. We can see the results in graphs for 200 measurements (Figure 7.13), 400 measurements (Figure 7.14) and 800 measurement (Figure 7.15). The number represents positions of the key bytes in correlation matrix in total. For instance, when all the bytes of the key are guessed correctly, the total number is 16, but when one byte is in the second position and the others are in the first position, the total number is 17. The less the total number is, the better the results are. The highest position of a byte is 256 because we are

guessing bytes and a byte can only have 2^8 different values.

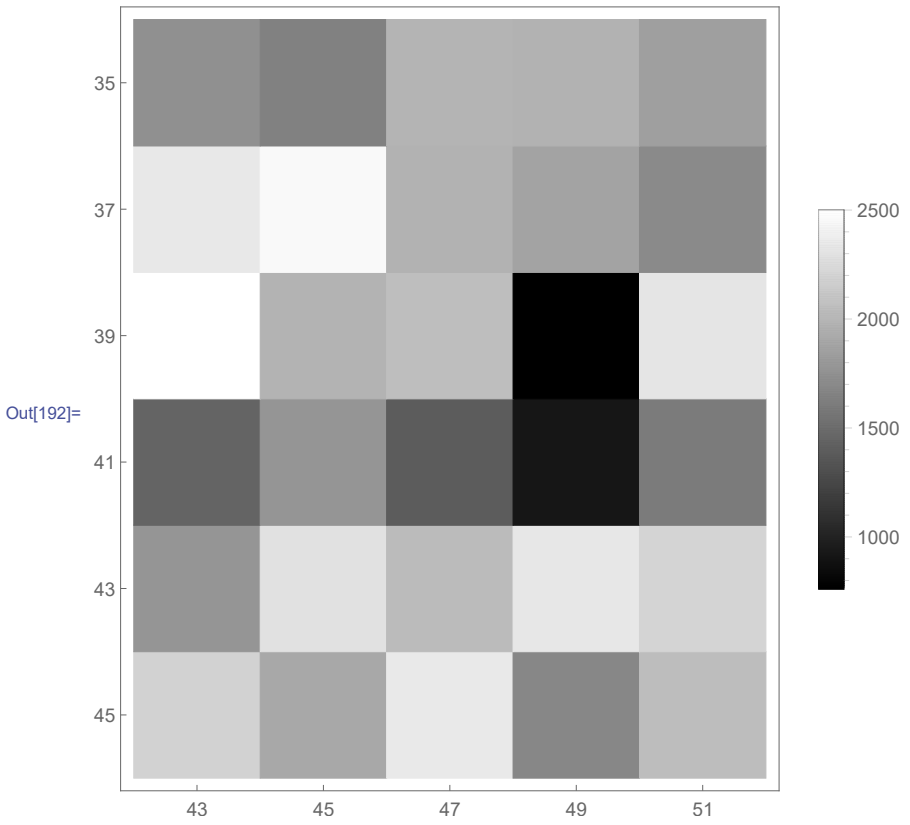


Figure 7.13: Sum of positions of all correct key bytes in lists sorted by the correlation coefficient, 200 measurements

In the graph for 200 measurements (Figure 7.13), the lowest total number of positions was 762 in the position [49,39], which leads to an average position 47.6 in the correlation matrix. The second lowest number is 913 in the position [49,41] which is the position about 56 on average in the correlation matrix. We can see that the lowest number does not need to fully correspond to the highest number of correctly guessed bytes. Even though the average position in coordinates [49,39] was lower, the number of bytes guessed in coordinates [49,41] was higher. In all coordinates the positions in the correlation matrices are high and it is evident that we need more measurements to be able to attack the smart card.

The results in the graph for 400 measurements (Figure 7.14) are much better than the results for 200 measurements. The two lowest total numbers of positions are at the same coordinates as for the 200 measurements. At the coordinates [49,39] the total number is 107 and at the coordinates [49,41]

is 240. The second lowest number is more than a double of the first lowest number, even though at the coordinates [49,41] ten bytes of the cipher key were guessed correctly and at the coordinates [49,39] nine bytes were guessed correctly. The third lowest number was 550 at the coordinates [51,41], where 6 bytes were guessed correctly. The fourth one was [47,39], where 3 bytes were guessed correctly. Except for the first two positions, the total number of the positions corresponds to the number of correctly guessed bytes of the cipher key.

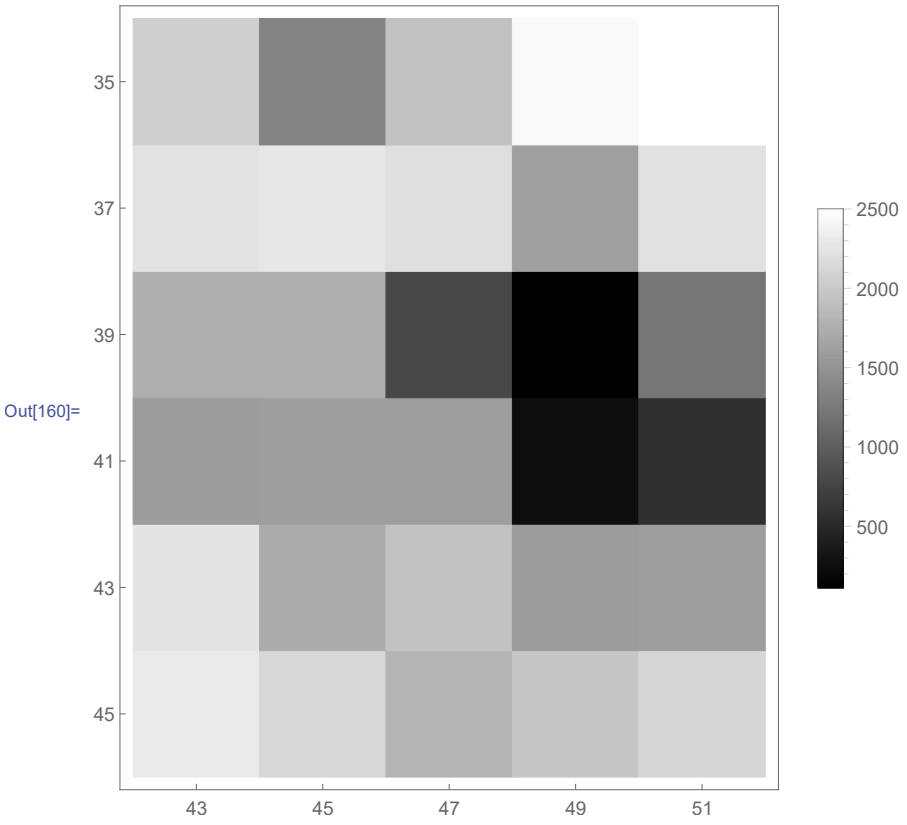


Figure 7.14: Sum of positions of all correct key bytes in lists sorted by the correlation coefficient, 400 measurements

The best results are for 800 measurements (Figure 7.15). From this graph it is obvious that the middle part of the Y-axis yields the best results. We have three best positions where we guessed all the bytes correctly: [49,39], [49,41] and [51,41]. All the results in the Y coordinate 41 and 39 were low. Taking the whole chip, there are still places that are hard to attack even for 800 measurements, the worst position is [45,37] with total number 2089, even though this position is close to the centre of the chip.

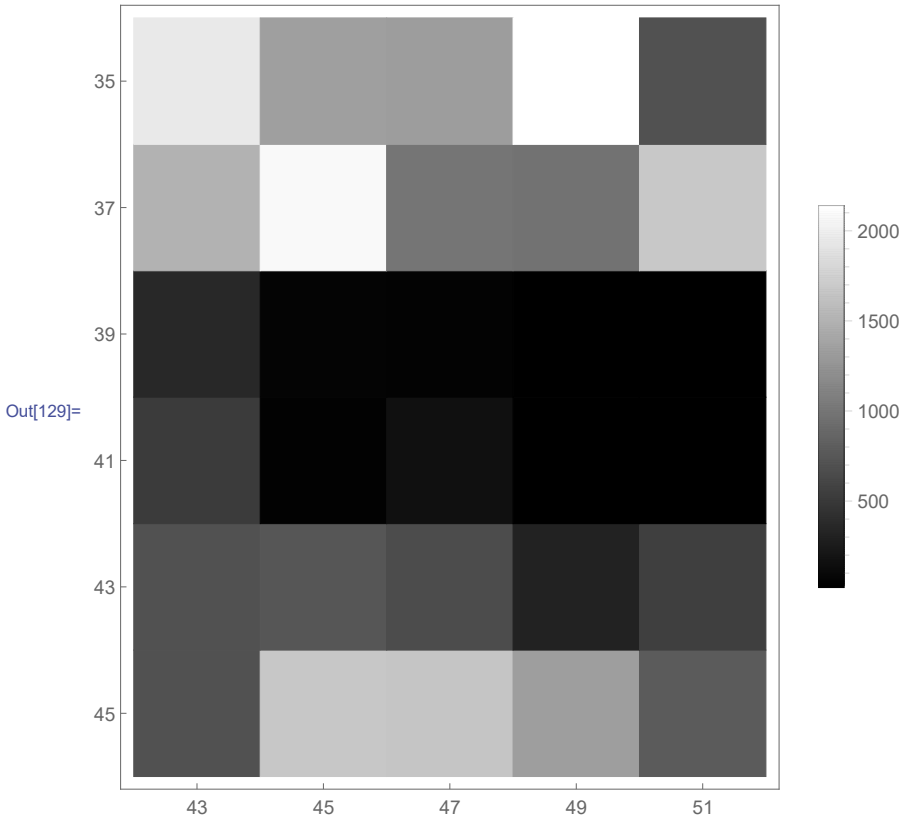


Figure 7.15: Sum of positions of all correct key bytes in lists sorted by the correlation coefficient, 800 measurements

7.3 3D Positioning

We did a similar kind of measurements in three-dimensional space as in the two-dimensional space. We measured mean of the signal in different positions. This time, we measured each millimetre. We covered the area above the chip. The size of the measured area was 10 mm x 12 mm x 6 mm. The mean of the values did not differ much, the difference was only around 5 %. We can see the measured intensity of the radiation in Figure 7.16. There are two major places with the highest intensity. The places are in the coordinate Z=1, coordinates X and Y are [45-47, 41-42] and [45-47, 38-39]. The strongest signal was at the coordinates [46,39]. These position are around the centre of the chip. We can see that the rest of the signal has the similar intensity and it does not matter what the Z coordinate is.

We did 40 measurements in each position. We could guess at most three bytes of the key correctly. We calculated the least bytes of the cipher key

correctly at the first layer, which could appear strange. We calculated only one byte correctly in three positions and the positions were at the borders of the chip. In the second layer we calculated one byte correctly at six positions and two bytes correctly at two positions. Since we only conducted 40 measurements in each position, the correctly guessed bytes can be only a coincidence and does not have to mean that if we put the probe higher, we will get better results.

For estimating the best spot for guessing the most bytes, we would have to do more measurements in each position. We did not do it in this thesis, because this mapping requires a lot of time. In three-dimensional space we have to decide what the best spot for an attack is based on the mean of the radiation intensity.

Figure 7.16: Graph of signal intensity in three-dimensional space

Conclusion

The primary purpose of this thesis was to build a three-dimensional near-field acquisition system, basically to create a positioning device that can position a probe or an antenna in different locations in three-dimensional space. Another goal was to design and to implement a program that can communicate with the built positioning device, a measuring equipment (oscilloscope) and a smart card. The next part was implementing a program in C++ for calculating DEMA using the measured data. The testing part consists of measuring electromagnetic radiation in various locations of a smart card's chip. After measuring the emissions, the acquired data are analysed and the best spot for an attack is found. The last goal was to make graphs using measured electromagnetic emissions and differential electromagnetic analysis results.

We managed to accomplish all the aims of this thesis. To achieve them, we built a positioning device using 3D printing technology. We connected a Minitronics board to the stepper engines and programmed it. We printed holders for the probes that we used for measuring electromagnetic radiation. We attached a smart card reader to the positioning device's bed. We also fastened the probe's holders to the positioning device. After that, we executed measurements in one place, in two-dimensional and three-dimensional space. We analysed the results and made graphs out of them. Firstly, we computed DEMA in one spot only; we discovered that we needed at least 400 measurements to guess the whole cipher key correctly. Then we continued with two-dimensional and three-dimensional mapping. We found two locations with the highest mean values of the electromagnetic radiation. We discovered that the spots with the strongest electromagnetic radiation are not necessarily the places where we can guess the most bytes of the cipher key correctly. The signal at the locations with the most successful DEMA was average and sometimes even below average. We mapped the three-dimensional space around the chip; it is evident that radiation right above the chip is stronger than the radiation further from the chip.

In the future, we can use three-dimensional acquisition system for mapping signal radiating from other devices, e.g. an FPGA. We can use this device to get the best positions for an attack. The positioning device controlled by the implemented program can analyse the measured data automatically, and the measurements are repeatable. The results can be later used for making these devices more secure against electromagnetic side-channel attacks.

Bibliography

[1] EMV-Technik, L. *RF2 Near Field Probe Set*. Accessed: April 7, 2018. Available from: https://www.langer-emv.de/en/product/rf-passive-30-mhz-3-ghz/35/rf2-set-near-field-probes-30-mhz-up-to-3-ghz/272

[2] Neruda, M. *Návod na stavbu 3D tiskárny Rebelix*. Accessed: May 1, 2018. Available from: http://reprap4u.cz/navod-na-stavbu-3d-tiskarny-rebelix/

[3] D'Albore, A. What is a Smart Card. 2017, accessed: April 10, 2018. Available from: https://embeddedsecuritynews.com/2017/02/what-is-a-smart-card

[4] Ortiz, C. E. An Introduction to Java Card Technology - Part 1. 2003, accessed: April 10, 2018. Available from: http://www.oracle.com/technetwork/java/javacard/javacard1-139251.html

[5] RepRapProject. Minitronics 10. Accessed: May 7, 2018. Available from: http://reprap.org/wiki/Minitronics_10

[6] Kuhn, M. G.; Anderson, R. J. *Information Hiding*, chapter Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations. Springer-Verlag Berlin Heidelberg, 1998, pp. 124–142.

[7] Friedman, J. Tempest: A signal problem. *NSA Cryptologic Spectrum*, 1972.

[8] Singel, R. Declassified nsa document reveals the secret history of tempest. *Wired Magazine*, 2008.

[9] Eck, W. V. Electromagnetic radiation from video display units: An eavesdropping risk? In *Computers & Security*, 1985, pp. 269–286.

[10] Agrawal, D.; Archambeault, B.; et al. The EM Side—Channel(s). In *Cryptographic Hardware and Embedded Systems - CHES 2002*, edited by B. S. Kaliski; ç. K. Koç; C. Paar, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, ISBN 978-3-540-36400-9, pp. 29–45.

[11] De Mulder, E. Electromagnetic techniques and probes for side-channel analysis on cryptographic devices. In *Diss. PhD Thesis*, Kasteelpark Arenberg 10, B-3001 Leuven (Belgium): Katholieke Universiteit Leuven, Faculty of Engineering, 2010, ISBN 978-94-6018-281-5.

[12] Maxwell, J.; Thompson, J. *A Treatise on Electricity and Magnetism.* Number sv. 2 in A Treatise on Electricity and Magnetism, Clarendon, 1904. Available from: `https://books.google.cz/books?id=t5vCDCXPUswC`

[13] Lenz, H. F. E. *Annalen der Physik und Chemie*, chapter Ueber die Bestimmung der Richtung der durch elektodynamische Vertheilung erregten galvanischen Strome. 1834, pp. 483–494.

[14] National Institute of Standards and Technology. Advanced Encryption Standard. *NIST FIPS PUB 197*, 2001.

[15] Mangard, S.; Oswald, E.; et al. Differential Power Analysis. In *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, Boston, MA: Springer US, 2007, ISBN 978-0-387-38162-6, pp. 119–165, doi:10.1007/978-0-387-38162-6_6, accessed: March 20, 2018. Available from: `https://doi.org/10.1007/978-0-387-38162-6_6`

[16] Ding, G.-l.; Li, Z.-x.; et al. Differential Electromagnetic Analysis on AES Cryptographic System. In *2009 Second Pacific-Asia Conference on Web Mining and Web-based Application*, June 2009, pp. 120–123, doi:10.1109/WMWA.2009.46.

[17] Réal, D.; Valette, F.; et al. Enhancing Correlation Electromagnetic Attack Using Planar Near-field Cartography. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2009, ISBN 978-3-9810801-5-5, pp. 628–633, accessed: April 2, 2018. Available from: `http://dl.acm.org/citation.cfm?id=1874620.1874777`

[18] Briggs, T. Introduction to Oscilloscopes. Accessed: April 17, 2018. Available from: `https://beyondmeasure.rigoltech.com/acton/attachment/1579/f-06a8/1/-/-/-/-/EDU%20Basic%20Lab%20Guide.pdf`

[19] Keysight. *Keysight InfiniiVision, 3000 X-Series Oscilloscopes.* Keysight Technologies, Inc., 1900 Garden of the Gods Road, Colorado Springs,

CO 80907 USA, February 1, 2018, accessed: April 17, 2018. Available from: https://www.keysight.com/upload/cmc_upload/All/3000_series_prog_guide.pdf

[20] Rankl, W.; Effing, W. *Smart card handbook*. John Wiley & Sons, 2004.

[21] Advanced Card Systems Ltd. *ACR38U-I1 Smart Card Reader, Technical Specifications V1.12*. Accessed: May 2, 2018. Available from: https://www.acs.com.hk/en/products/199/acr38u-i1-smart-card-reader/

[22] Meijer, B. *Minitronics v1.1 Datasheet*. May 19, 2014.

[23] TexasInstruments. *DRV8825 Stepper Motor Controller IC*. Accessed: May 7, 2018. Available from: http://www.ti.com/lit/ds/symlink/drv8825.pdf

[24] Keysight. Software I/O Layers - VISA, VISA COM, SICL, Keysight 488 - Technical Overview. Accessed: May 8, 2018. Available from: https://www.keysight.com/main/editorial.jspx?cc=CZ&lc=eng&ckey=1461160&nid=-33330.977662.00&id=1461160

[25] Agilent. *Keysight Technologies, 6000 Series Oscilloscopes - Data Sheet*. January 21, 2005, accessed: April 17, 2018. Available from: https://literature.cdn.keysight.com/litweb/pdf/5989-2000EN.pdf

Appendix A

AES

As we wrote in section 1.2, AES is a symmetric block cipher. The key length can vary, it can be 128, 192 or 256 bits. The cipher uses the Rijndael algorithm. Firstly, a key expansion is conducted, then a loop is executed. The loop is executed for each round of each subkey. First of all, in each iteration, there is a byte substitution, then bytes are shifted, and columns are mixed. At the end of each iteration, the round key is added. After the end of the loop, there is one more byte transformation, rows shift and adding round key, but there is no more column combination.

A.1 Definitions

- N_c ... Number of columns (32-bit words)
- N_w ... Number of 32-bit words compromising cipher key
- K ... Key
- N_r ... Number of rounds
- round ... one iteration of the cycle of the cipher
- state array ... array representation of intermediate Cipher results
- word ... group of 32-bits
- plaintext ... cipher's input
- ciphertext ... cipher's output

N_k and N_r change according to the key length.

	AES-128	AES-192	AES-256
Block Size (Nb)	4	4	4
Key Length (Nk)	4	6	8
Number of rounds (Nr)	10	12	14

Figure A.1: AES

AES operations are internally executed on 2D fields. At the beginning of encryption or decryption, the input is copied to a state array and operations are executed on this array, then the array is copied to the output.

A.2 Key Expansion Algorithm

Key expansion uses the Rijndael algorithm. It takes cipher key K, and Key Expansion is executed to generate a Key Schedule. The algorithm needs N_r words, and each round needs N_b words of the key. In total algorithm generates:

$$N_b(N_r + 1) \tag{A.1}$$

words. Final key schedule consists of linear array of four bytes words $[W_i]$, where

$$0 \leq i < N_b(N_r + 1). \tag{A.2}$$

First N_k words are filled with the cipher key. After that, each word is equal to XOR of the previous word and a word that is N_k position before the current word. For words that are multiples of N_k, XOR with round constant Rcon[i] and transformation of the previous word is executed before.

A.2.1 Substitute words

This function is used to substitute bytes of the input. It takes four bytes and each byte is substituted using S-Box, see Table A.1.

A.2.2 Rotate Word

This function does cyclic permutation of four bytes word.

$$Input : [a0, a1, a2, a3]$$
$$Output : [a1, a2, a3, a0]$$

A.2.3 Pseudocode

Algorithm 1 Key Expansion

```
1:  procedure KeyExpansion(byte key[4 * Nk], word w[Nb *(Nr +1)], Nk)
2:      word tmp
3:      for i = 0    to    Nk − 1 do
4:          w[i] ← word(key[4 * i], key[4 * i + 1], key[4 * i + 2], key[4 * i + 3])
        end for
5:      for i = Nk    to Nb(Nr + 1) − 1 do
6:          tmp ← w[i − 1]
7:          if i mod Nk == 0  then
8:              tmp ← SubWord(RotWord(tmp)) ⊕ Rcon[i/Nk]
9:          else if Nk > 6 && i mod Nk == 4  then
10:             tmp ← SubWord(tmp)
11:         w[i] ← w[i − Nk] ⊕ tmp
12:         i+ = 1
        end for
```

A.3 Round

At the beginning of each round, the input is filled from left to right and from top to bottom by columns to 4×4 bytes matrix **A**. Then substitution is applied to each byte of the matrix **A**. Substitution is described in Table A.1. After that, there is operation *ShiftRows*, where lines of matrix **A** are cyclically shifted gradually over 0 to 3 bytes to the left. The first row is shifted over 0 bytes, second over 1 byte, third over 2 bytes and the fourth one over 3 bytes, this leads to byte transposition on bytes level. Operation *MixColumns* is then applied; it is a substitution of 32 bits to 32 bits, so each output bit is a linear combination of input bits. Finally new round key is added and operations *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey* are repeated according to the number of rounds. In the last round the operation *MixColumns* is not applied.

A.3.1 Pseudocode

Algorithm 2 AES

```
1: procedure CIPHER(byte in[4 * N_k], byte out[4 * N_b], word w[N_b *(N_r+1)])
2:     Byte A[4, N_b] ← matrix for temporary results
3:     A ← in
4:     AddRoundKey(state, w[0, Nb-1]
5:     for round = 1  to  round = N_r do
6:     SubBytes(state)
7:     ShiftRows(state)
8:         if round ≠ N_r  then MixColumns(state)
9:     AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
10:    end for
11: return state
```

A.3.2 Add Round key

Each round key consists of N_b words that are created during key expansion. At the beginning before the first round, there is an initial function, where the first four round keys are XORed to plaintext (128 bits on 128 bits).

A.3.3 Word substitution

This function takes four bytes and applies S-Box (Table A.1) to each byte. S-Box is constructed using a calculation of inverse multiplicative element of each byte in finite field GF(2^8) with the irreducible polynomial

$$m(x) = x^8 + x^4 + x^3 + x + 1. \tag{A.3}$$

Neutral element (the element that is transformed to itself) is 00, after that an affine transformation is applied, where

$$b_i = b_{(i+4)} \mod 8 \oplus b_{(i+5)} \mod 8 \ldots \oplus b_{(i+7)} \mod 8 \oplus c_i \tag{A.4}$$

$$c_i = i - t_7 \quad = \quad 0. \qquad \text{bit from byte c, where c=63} \tag{A.5}$$

S-Box is represented with hexadecimal numbers in the substitution table (Table A.1).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Table A.1: SBox

A.3.4 Shift Rows

Bytes in the last three rows of the State table are cyclically shifted by a given number of bytes (offset).

$$S'_{r,c} = S_{r,(c+shift(r,Nb))modN_b} \qquad for \quad 0 < r < 4 \quad and \quad 0 \leq c \leq Nb$$

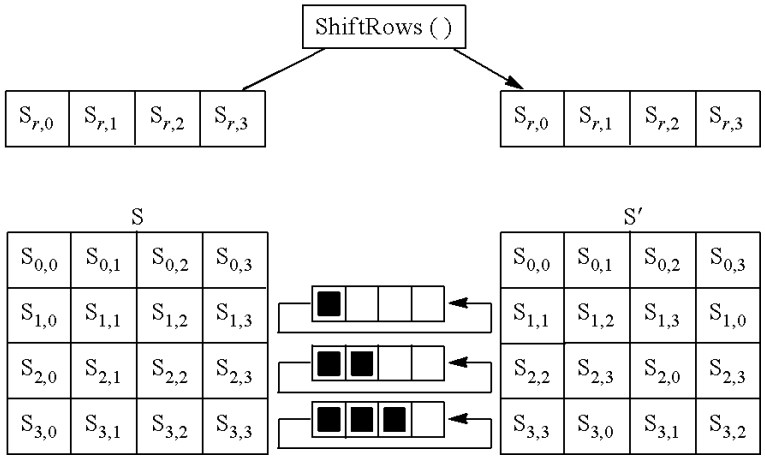


Figure A.2: Shift Rows

A.3.5 MixColumns

Function is applied column by column; it takes each column as a four-element polynomial. Columns are considered to be polynomials over a finite field

GF(2^8) and multiplied by a fixed polynomial $a(x)$ modulo $x^4 + 1$.

$$a(x) = 03x^3 + 01x^2 + 01x + 02 \tag{A.6}$$

$$S'(x) = a(x) \otimes S(x) \tag{A.7}$$

$$\begin{aligned}
S'_{0,c} &= (02 \bullet S_{0,c}) \oplus (03 \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c} \\
S'_{1,c} &= S_{0,c} \oplus (02 \bullet S_{1,c}) \oplus (03 \bullet S_{2,c}) \oplus S_{3,c} \\
S'_{2,c} &= S_{0,c} \oplus S_{1,c} \oplus (02 \bullet S_{2,c}) \oplus (03 \bullet S_{3,c}) \\
S'_{3,c} &= (03 \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (02 \bullet S_{3,c})
\end{aligned}$$

Appendix B

Positions of the Correct Key in the Correlation Matrices

Table B.1 shows position in the correlation matrix of the correct cipher key for 200 measurements. Table B.2 shows the results for 300 measurements and Table B.3 shows the results for 400 measurements.

Byte\Run	1	2	3	4	5	6	7	8	9	10	Correctly Guessed
1	209	1	60	34	8	20	1	1	1	144	4
2	1	1	1	1	112	1	1	5	8	1	7
3	1	57	1	8	86	1	1	15	26	3	4
4	4	1	1	3	2	4	2	1	1	1	5
5	1	7	12	1	76	65	2	4	1	2	3
6	7	1	19	1	15	1	138	2	1	11	4
7	59	1	7	1	5	121	15	70	1	11	3
8	2	1	3	1	2	3	13	1	4	32	3
9	10	3	45	71	100	28	9	12	3	2	0
10	1	1	9	1	123	37	12	16	6	37	3
11	1	1	1	65	134	11	1	1	4	1	6
12	5	16	5	2	2	3	10	3	24	6	0
13	1	63	4	34	8	24	249	221	101	125	1
14	3	3	15	135	5	7	75	38	85	128	0
15	1	1	1	1	1	1	16	2	1	1	8
16	1	1	1	2	1	1	32	16	6	93	5
Correctly Guessed	8	10	6	7	2	5	4	4	6	4	

Table B.1: Position in the correlation matrix of the correct key of each byte of cipher for each run for 200 measurements

Byte\Run	1	2	3	4	5	6	7	8	9	10	Correctly Guessed
1	13	1	1	2	4	1	5	3	3	1	4
2	1	1	1	1	5	1	25	1	1	1	8
3	5	1	1	6	4	1	1	2	1	1	6
4	1	1	1	1	2	1	1	1	1	1	9
5	21	1	1	1	1	8	1	2	1	2	6
6	1	2	1	1	2	3	1	1	4	1	6
7	1	1	1	1	1	1	1	1	1	1	10
8	1	1	8	1	25	1	1	1	1	1	8
9	1	3	1	7	1	1	1	1	1	1	8
10	1	4	1	1	2	1	1	1	1	1	8
11	1	7	1	1	1	2	1	1	1	1	8
12	2	1	1	1	1	5	1	1	1	1	8
13	1	2	1	36	1	21	44	1	85	4	4
14	1	1	1	1	1	1	1	1	1	1	10
15	1	1	1	1	1	1	1	1	1	1	10
16	1	1	1	1	2	1	1	2	1	2	7
Correctly Guessed	12	11	15	12	8	11	13	12	13	13	

Table B.2: Position in the correlation matrix of the correct key of each byte of cipher for each run for 300 measurements

Byte\Run	1	2	3	4	5	6	7	8	9	10	Correctly Guessed
1	1	1	2	1	1	1	1	1	1	1	9
2	1	1	1	1	1	1	1	1	1	1	10
3	1	1	1	1	1	1	1	1	1	1	10
4	1	1	1	1	1	1	1	1	1	1	10
5	4	1	1	1	1	1	1	1	2	1	8
6	1	1	1	1	1	1	1	1	1	1	10
7	1	1	1	1	1	1	1	1	1	1	10
8	1	2	1	1	1	2	1	1	1	1	8
9	1	1	6	1	1	1	1	1	1	1	9
10	1	1	1	1	1	3	1	1	1	1	9
11	1	1	1	1	1	1	1	1	1	1	10
12	1	1	1	1	1	4	1	1	1	1	9
13	1	1	3	1	1	1	1	1	1	2	8
14	1	1	1	1	1	1	1	1	1	1	10
15	1	1	1	1	1	1	1	1	1	1	10
16	1	1	1	1	1	1	1	1	1	1	10
Correctly Gussed	15	15	13	16	16	13	16	16	15	15	

Table B.3: Position in the correlation matrix of the correct key of each byte of cipher for each run for 400 measurements

Appendix C

Acronyms

ADS Analog-to-Digital Converter

AES Advanced Encryption Standard

APDU Application Protocol Data Unit

API Application Programming Interface

ASM Assembler Source Language

CEMA Correlation Electromagnetic Analysis

CPA Correlation Power Analysis

CNC Computer Numeric Control

DEMA Differential Electromagnetic Analysis

DES Data Encryption Standard

DPA Differential Power Analysis

EEPROM Electrically Erasable Programmable Read-Only Memory

EMA Electromagnetic Analysis

EMSEC Emission Security

FIPS Federal Information Processing Standards

ISO International Organization for Standardization

LAN Local Area Network

RFID Radio-frequency identification

ROM Read-Only Memory

SCPI Standard Commands for Programmable Instruments

SEMA Simple Electromagnetic Analysis

SPA Simple Power Analysis

STL Standard Template Library/Stereolithography

USB Universal Serial Bus

VISA Virtual Instrument Software Architecture

Appendix D

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `impl` ..... the directory of the implemented program
    - `documentation` ..... the directory of documentation of the sources
    - `program` ..... the directory of sources of the implemented program
  - `thesis` ..... the directory of LaTeX source codes of the thesis
  - `Marlin-RC` ..... the Marlin firmware for Minitronics
  - `AES` ..... the AES implementation for a smart card
- `graphs` ..... the directory of created graphs
  - `data` ..... the directory of the source data for the graphs
  - `graphs.nb` ..... the file with graphs
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format