UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Side-Channel Analysis of a Compact AES FPGA Implementation

Seitenkanal-Analyse einer kompakten AES FPGA Implementation

Bachelorarbeit

im Rahmen des Studiengangs
Informatik
der Universität zu Lübeck

vorgelegt von
Oguzhan Tekin

ausgegeben und betreut von
Prof. Dr. Thomas Eisenbarth

mit Unterstützung von
Okan Seker, M. Sc.

Lübeck, den 01. November 2019

Erklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Lübeck, 01. November 2019

Abstract

Today, applications are used by all people worldwide. However, they require comprehensive protection against external interference on every device. Therefore, the security of these implementations must be mathematically guaranteed. The security of electronic devices is often based on cryptographic algorithms. These algorithms have proven mathematically reliable and are therefore virtually unbreakable. Unfortunately, the developers only guarantee the security of such devices cryptographically and there are types of attacks that are hardly or only partially considered. One possible type of attack is the side channel attack (SCA), which has attracted a lot of attention in the community in recent years. Field Programmable Gate Arrays are of particular interest today, as they are an attractive choice for use in cryptographic devices, but they prove to be a major weakness in SCA. For this purpose we show the weakness of these devices by implementing a compact AES-128 algorithm to an FPGA board and process a performance and security analysis. We perform well-known side channel attacks like, differential power analysis, correlation power analysis and template attacks.

Zusammenfassung

Heute werden Anwendungen von allen Menschen weltweit genutzt. Sie erfordern jedoch einen umfassenden Schutz vor äußeren Einflüssen auf jedes Gerät. Daher muss die Sicherheit dieser Implementierungen mathematisch gewährleistet sein. Die Sicherheit von elektronischen Geräten basiert oft auf kryptographischen Algorithmen. Diese Algorithmen haben sich mathematisch bewährt und sind daher nahezu unzerbrechlich. Leider garantieren die Entwickler die Sicherheit solcher Geräte nur kryptographisch und es gibt Arten von Angriffen, die kaum oder nur teilweise berücksichtigt werden. Eine mögliche Angriffsart ist der Seitenkanalangriff (SCA), der in den letzten Jahren in der Community große Aufmerksamkeit erregt hat. Field Programmable Gate Arrays sind heute von besonderem Interesse, da sie eine attraktive Wahl für den Einsatz in kryptographischen Geräten sind, sich aber als eine große Schwäche bei Seitenkanalangriffen erweisen. Zu diesem Zweck zeigen wir die Schwäche dieser Geräte auf, indem wir einen kompakten AES-128-Algorithmus auf einem FPGA-Board implementieren und eine Leistungs- und Sicherheitsanalyse durchführen. Wir führen bekannte Seitenkanalangriffe wie Differential-Power-Analyse, Korrelations-Power-Analyse und Template-Angriffe durch.

Contents

1 Introduction

1.1 Cryptography

Cryptology is a science that deals with the encryption and decryption of information, information security and the analysis of such systems. It can be divided into two main component: Cryptography and cryptanalysis [MVO96]. First component, cryptanalysis is the study of mathematical techniques for attempting to break cryptographic techniques and systems. The goal of cryptanalysis in general is to decrypt encrypted messages without knowing the key or even to find out the secret key. Secondly, cryptography is a part of cryptology whose task is to construct encryption methods that are secure in terms of information technology. These methods are mathematical functions which receive a plaintext and a key as input parameters and return a ciphertext as output depending on the plaintext.
Today, cryptographic methods are used to guarantee four aspects in security systems which can be summarized as follows: [MVO96].

- **Confidentiality:** Confidentiality is a service to protect information content from those who do not have access to it.
- **Data integrity:** Data integrity is a service to protect information from manipulation or modification by unauthorized persons. In addition, it must also be ensured that any unauthorised changes to the data can be traced or detected.
- **Authentication:** Authentication is a service for identifying at least two parties during communication. It must be ensured that a communication partner can identify another partner so that no unauthorized person can impersonate one of the parties.
- **Non-repudiation:** Non-repudiation is a service that prevents the rejection of previous acts and duties.

There are two types of encryption schemes: Symmetric and asymmetric encryption. Symmetric encryption methods are methods that use the same key for encryption and decryption. This also means that communication between two or more communication partners requires a single key that has been securely exchanged prior to communication. There are procedures such as the Diffie-Hellmann [Buc16] for the secure exchange of a key.

Asymmetric encryption methods are methods which, in contrast to symmetric methods, require key pairs for the encryption and decryption of information. A key pair consists of a public key and a secret key. Both keys of a pair are created dependent on each other. The secret key is used to decrypt information that has been encrypted with its public key. When two partners communicate, both have their own key pairs. To send a message, it must be encrypted with the recipient's public key. The recipient must now use his secret key to read the message. A well-known method of asymmetric cryptography is the RSA encryption method [Buc16].
For the further course of the work only one known symmetric procedure is of importance: Advanced Encryption Standard. In 1997, the National Institute of Standards and Technology (NIST) called for the development of a symmetric cryptography technique and announced the following conditions: Variable key length of 128 bits, 192 bits, 256 bits and a fixed block length of 128 bits. It was intended to replace the Data Encryption Standard (DES) already existing at the time. A competition was called. In the end, there were five algorithms to choose from, which were subjected to precise testing. In 2000, an encryption method was chosen as the standard under the name Advanced Encryption Standard (AES), developed by the Belgian cryptologists Daemen and Rijmen under the name Rijndael. At that time DES was replaced by AES.
Advanced Encryption Standard (AES) is an encryption algorithm that encrypts in blocks. The same key is used for encryption and decryption. The algorithm has key lengths of 128, 196 or 256 bits, which can be selected variable and performs in 10, 12 or 14 rounds respectively.
The AES procedure performs the following operations: Initially, all round keys are calculated using the **KeyExpansion** operation. The **AddRoundKey** operation is performed at the beginning of the first round and at the end of each round. The current block is encrypted with the respective round key. In each round **SubBytes** is executed, which applies the S-Box operation to all bytes. The **ShiftRows** operation shifts the rows of a block and is also applied to each round. The last operation **MixColumns** mixes the bytes within the columns. This operation is performed in all rounds except the last round.

1.2 Physical Attacks

Physical attacks in cryptography describe two classes of attacks; the attacks that utilize the passive side channel information like power [KJJR11], electromagnetic emission [GMO01] or acoustic [GST14] of cryptographic devices and second actively manipulating the device, for example the manipulation of the implementation. [LP05]. There are mainly five types of physical attacks: intrusion, monitoring, manipulation, modification, and substitution.

The work will be mainly about monitoring. Monitoring is a passive attack in which no changes are made to the device or implementation. Only measurements of electromagnetic emanations or energy consumption are performed. One attack that will be the main focus of this work is the side channel analysis. These attacks are passive attacks based primarily on monitoring. Two types of attacks are presented in the paper [KJJR11].

1.3 Power Analysis Attacks

Power Analysis Attacks are physical attacks on cryptographic devices. This type of physical attack aims to determine the cryptographic key of a device by measuring its power consumption. Usually two dependencies of the device are exploited: Data dependency and power consumption dependency. When operating the device, it is exploited that the current power consumption of the device depends on the processed data and processes.
There are two main types of Power Analysis Attacks: Simple Power Analysis and Differential Power Analysis. The work mainly deals with the experimental implementation of Differential Power Analysis.
Simple Power Analysis (SPA) is a type of physical attack in which the attacker derives the key from side channel information. The power consumption caused by the cryptographic operations is measured and interpreted directly. In practice, the technique is very complex. The implementation requires knowledge about the implementation, i.e. about the structure of the algorithm. These attacks can be used to analyze fewer traces [MOP07].
Differential Power Analysis (DPA) is the most popular type of power analysis attack. In this attack hardly any knowledge about the attacked device is necessary. For this attack requires a large number of power traces. Therefore, it is also necessary that the attacker has physical access to the attacked device, even if only for a short time. The goal is the same as with SPA: To recover the cryptographic key [MOP07].

1.4 Cryptographic Devices

Cryptographic devices are electronic devices that execute algorithms on them. They consist of several components, each of which has an existing functionality. For example, a component that is included in all the cryptographic devices has the function of performing cryptographic operations. These are generally known as microcontrollers. Another component is the memory that stores the data required for the operations.
For communication to and from the board a component exists as an interface. This is responsible for the transfer of data.
There are different variants of cryptographic devices such as USB sticks, smart cards or Field Programmable Gate Arrays (FPGAs). The types of attacks mentioned above can be

carried out on these devices as well as on other physical devices that are not listed. For further work only FPGAs will be of importance.
A Field Programmable Gate Array (FPGA) is an integrated circuit and a logical device in which any digital circuits can be implemented. The basic structure of an FPGA is an array of blocks that have a lookup-table, which can be denote as an arbitrary gate, and flip-flops. In addition, an FPGA has an input and an output block, which receives signals from outside and sends outward. For FPGAs, unlike microcontrollers and computers, the focus is not only on timing, but also on the desired design of the circuit. For the implementation, a hardware description language is used and converted into a file by means of software, which describes how the elements in the FPGA must be connected [Ama18].

1.5 Motivation

Today, the secure exchange of information plays an important role. Mathematically secure algorithms are used to secure such information. When exchanging large quantities, hardware implementations are used for efficiency. Hardware such as Field Programmable Gate Arrays are a preferred choice because of their flexibility and the efficient runtime through parallel work. At the end of the 20th century, it was discovered that such devices could provide leak information through physical access by attackers. Side channel attacks allow attackers to access information about the secret parameters of encryption algorithms. For some time it was assumed that devices such as FPGAs could prevent such leaks by processing them in parallel. Later, however, this thesis was refuted by numerous papers and works.

1.6 Outlook On The Work

The next chapters are as follows: First the attacks are introduced. It explains in general what the attacks do, what the target is and how they work. The focus of the chapter is mainly on Differential Power Analysis. Different variants of the DPA are explained. Next, FPGAs are introduced and explained in general. It describes how FPGAs are built and what programming possibilities are available in general. The following chapter describes our own implementation. First the used encryption method is introduced and the individual operations of it are described. Then the structure of the implementation and the communication with the board is explained. In addition, the properties of the implementation are enumerated and a comparison with other known implementations is made. The following chapter describes the execution of the experiment. The hardware used for this is presented. Then it is told how the measurement of the energy consumption took place. Subsequently, the analysis is described in detail and the results are presented by means of graphics. The last chapter serves as a conclusion of the work. There it is once again described what was done now, which methods were the most effective and which goal was in focus.

2 Side Channel Analysis

2.1 Simple Power Analysis

In this section we explain the basic principle of SPA attacks. There are two types of SPA attacks: single-shot SPA attacks and multiple-shot SPA attacks [MOP07]. With single-shot SPA attacks, only one track can be recorded, while multiple-shot SPA attacks can record multiple tracks. It can be decided whether the track is recorded several times for the same plaintext or whether only one track is recorded for different plaintexts. The advantage of repeating a plaintext multiple times is that it can reduce noise by finding the mean of the recordings. In both attacks, the principle is the same. The attacker uses collected traces to find the key.
The success of Simple Power Analysis in detecting keys is based on the following security problem: An implementation on a device is controlled by the microcontroller. The microcontroller has a number of arithmetic operations (for example, jump, shift, AND, XOR, OR, etc.). These operations work with a number of bytes and the implementation works only with the given arithmetic operations of the microcontroller. On the device, various components are created by these operations, which are physically separated from each other. Their functionalities are different. As a result, the components cause different consumption. An attacker who knows the implementation can analyze when measuring which component is working and thus can distinguish it from the power traces. Analyzing the order can be a big security issue because the key depends on the sequence of these operations.
One type of SPA attack are template attacks [CRR03]. Template attacks usually consist of two phases. In the first phase the analysis is carried out. In the second phase, the information obtained is used for an attack. The attack can be carried out as follows: A device of the same type is used. It executes command sequences and measures side channel information. The obtained traces are grouped and averages are determined. These values are used to analyze the values of the attacked device. This characterization attempts to determine the key on the attacked device. The analysis on the attacked device is carried out with different templates. The template with the highest probability is the correct template [MOP07]. SPA attacks have often been performed in the literature on AES implementations [Man03]. Likewise, these attacks were applied to the asymmetric cryptosystems like RSA [Nov02].

2.2 Differential Power Analysis

Differential Power Analysis (DPA) is a much more powerful attack then Simple Power Analysis (SPA) [KJJ98]. DPA uses statistical analysis techniques to obtain information about the secret key. The analysis only requires knowledge of the algorithm that runs on the device. Since no detailed knowledge of the attacked cryptographic device is required, DPA attacks are the most commonly used side-channel attacks in the literature ([MPL$^+$11],[BCO04],[CCD00],[CRR03]).
The goal of DPA is to uncover the cryptographic key that used during the encrypting or decrypting data blocks in a cryptographic system by recording a large number of side-channel information. These traces are analyzed in different ways during SPA and DPA attacks. While the SPA deals along the time axis, the DPA checks the dependency of the traces on the data at specified times. Since the dependency of the data and traces plays an important role here, a large number of power traces is advantageous. The more encryptions or decryptions are performed and recorded, the higher the probability that the key will be uncovered.

2.2.1 Execution Of DPA

In this work we focused on the DPA using known plaintexts. The version that uses known ciphertext can be defined in a similar way. The execution of DPA attacks can be performed in a few steps (Figure 2.1):

- First, random n plaintexts must be generated, which are known to the attacker. These are encrypted with an unknown key using the encryption algorithm within the device. The trace of the cryptographic device is measured while encrypting n different data blocks. During the measurement, the attacker receives the side channel trace $(t_{i,1}, \dots, t_{i,m})$ for the data block p_i, where m is the sample size of a trace. The traces of each data block can be stored as a matrix of the size $n \times m$.
- In the second step all possible keys are stored in a vector $(k_1, \dots, k_s)$. The vector is called a key hypothesis. Now the hypothetical intermediate values are calculated. The values are calculated by using $(p_1, \dots, p_n)$ and each possible $(k_1, \dots, k_s)$, where s represents the number of possible key candidates. That is, the attacker gets the hypothetical intermediate values as a matrix c of size $n \times s$ in such a way that the j^{th} column $(c_{1,j}, \dots, c_{n,j})$ corresponds to the j^{th} key candidate k_j.
- After the predictable intermediate values have been calculated, these values must be converted to hypothetical values. Thereby each predictable intermediate value $c_{i,j}$ is calculated as a hypothetical value $h_{i,j}$ and the attacker gets a matrix h of size $n \times s$

at the end. There are certain models with which the conversion can be performed, such as Hamming weight, Hamming distance or bit model.

- Finally, the power consumption values are compared with the trace. Every column i of trace $(t_{1,i}, \dots, t_{n,i})$ is compared with every column j of power consumption values $(h_{1,j}, \dots, h_{n,j})$. This compares the values of each key hypothesis with the real consumption values. As a result, the attacker now receives a matrix r of the size $s \times m$. The larger the values in matrix r, the greater the agreement of the key hypothesis with the real consumption values. After the analysis, the indexes of the largest value correspond to the key searched for.

For the match to occur, it must be assumed that the attacker has measured sufficient traces. The more traces the attacker has, the greater the probability that the key will be discovered. If there are not enough traces, an incorrect key may be detected.
Different statistical analysis methods exist for comparing the real consumption values with the hypothetical consumption values: Correlation-Based Analysis and Difference of Means Analysis.

2.2.2 Correlation-Based DPA

The Correlation-Based DPA, or Correlation Power Analysis (CPA), is a statistical analysis method for comparing two variables. It is a method that is very often used in DPA attacks. The pearson correlation is used for the comparison.

$$\rho = \frac{Cov(x, y)}{\sigma_x \sigma_y} \tag{2.1}$$

Pearson correlation coefficient is a method for measuring the linear relationship between two values or variables. The result of this measurement can only assume values between 1 and -1. The closer the value is to 1 or -1, the greater the agreement of the linear relationship. In Figure 2.1, σ is defined as the standard deviation and Cov is defined as the covariance.
The correlation coefficient is used in DPA attacks to perform the linear comparison of the columns of h and t described in section 2.2.1.

CPA Using Bit Model

The bit model is a power model that requires a single bit of a value. In Correlation-Based DPA, the first bit, the last significant bit (LSB), is usually used [MOP07]. The first bit usually contains the values required to calculate the next steps. Now hypothetical intermediate values are calculated from this power model. Finally, the correlation of the real

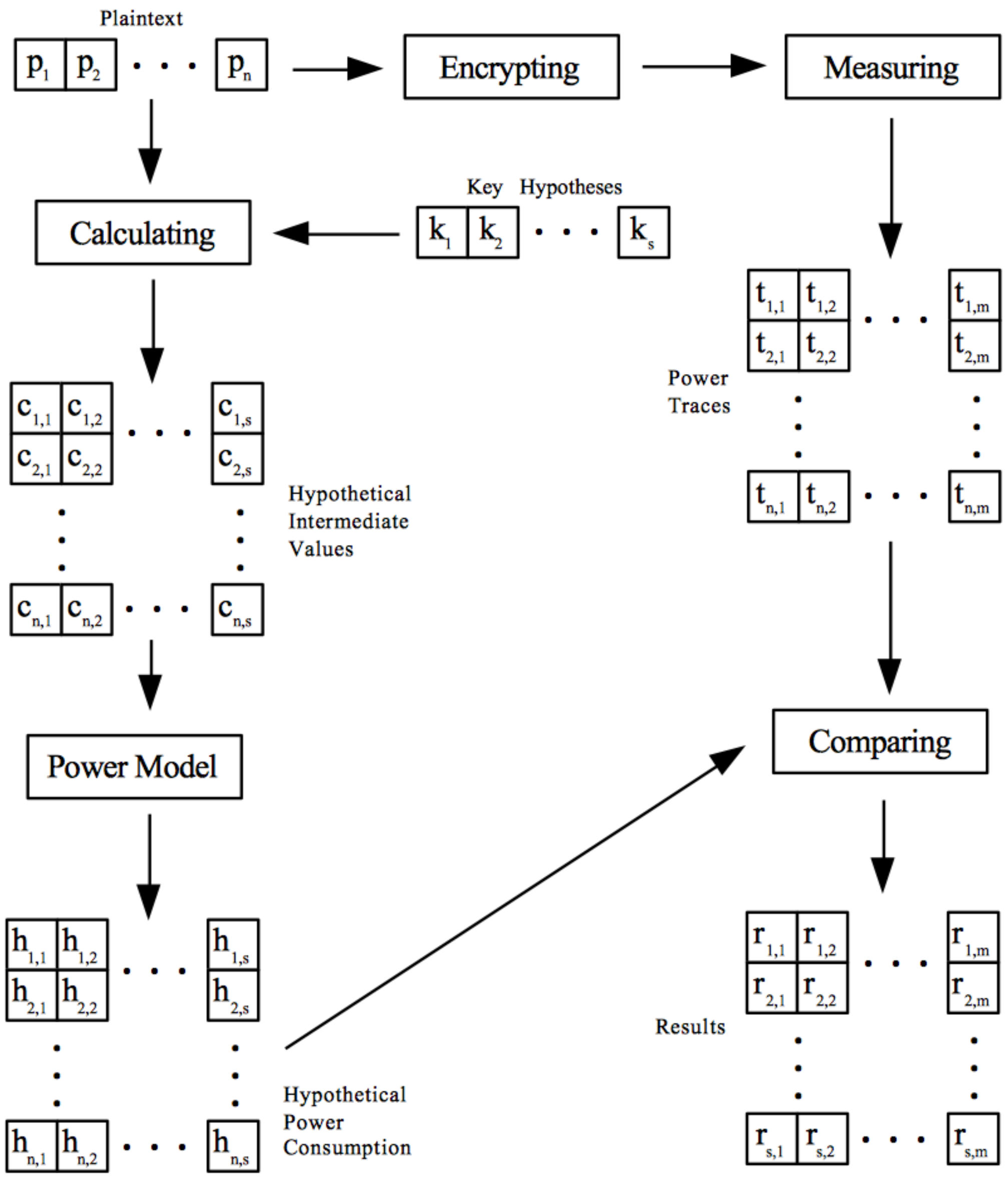

Plaintext
p_1 p_2 … p_n
Encrypting
Measuring
Key Hypotheses
k_1 k_2 … k_s
Calculating
Power Traces
$t_{1,1}$ $t_{1,2}$ … $t_{1,m}$
$t_{2,1}$ $t_{2,2}$ … $t_{2,m}$
$t_{n,1}$ $t_{n,2}$ … $t_{n,m}$
$c_{1,1}$ $c_{1,2}$ … $c_{1,s}$
$c_{2,1}$ $c_{2,2}$ … $c_{2,s}$
$c_{n,1}$ $c_{n,2}$ … $c_{n,s}$
Hypothetical Intermediate Values
Power Model
Comparing
$h_{1,1}$ $h_{1,2}$ … $h_{1,s}$
$h_{2,1}$ $h_{2,2}$ … $h_{2,s}$
$h_{n,1}$ $h_{n,2}$ … $h_{n,s}$
Hypothetical Power Consumption
Results
$r_{1,1}$ $r_{1,2}$ … $r_{1,m}$
$r_{2,1}$ $r_{2,2}$ … $r_{2,m}$
$r_{s,1}$ $r_{s,2}$ … $r_{s,m}$

Figure 2.1: Execution of DPA

consumption values with the hypothetically calculated ones is calculated. At the end, the column index that contains the absolute largest value is output as the correct key.

CPA Using Hamming Weight

The Hamming weight model is like the bit model a power model. The model is mostly used if the attacker has no information at all or if another power model does not produce results [MOP07]. The goal of the Hamming weight is to calculate the number of non-zeros. It is defined as the number of characters different from the null character.

After calculating the hypothetical intermediate values using the hamming weights, the correlation of the real consumption values with those of the hypothetical is calculated. The column index of the correlation matrix is the key that contains the largest value.

2.2.3 Difference Of Means

In the Difference of Means method, the relationship between the Hypothetical Power Consumption columns and the traces is determined by calculating the mean difference. The attacker assumes that certain values of a point in time generate a different consumption than the values at the other point in time. The columns of the binary matrix H are the calculated hypothesis values of all possible keys. The ones and zeros of a column basically determine the input function in the Difference of Means method. Therefore the traces are divided into two sets. The formula 2.4 and 2.5 shows us the calculation of the mean value of these two sets. The formulas 2.2 and 2.3 calculate the number of lines that are stored in the sets. Finally, the difference between $m1$ and $m0$ is calculated and stored in R (2.6). If the difference at a time is closer to, this means that the key hypothesis cannot be the correct key under any circumstances. If there is a large difference at a time, the key hypothesis is the right key to look for. Let us denote $m^1_{i,j}$ for the one set and $m^0_{i,j}$ for the zero set.

$$n_i^1 = \sum_{l=1}^{n} h_{l,i} \tag{2.2}$$

$$n_i^0 = \sum_{l=1}^{n} (1 - h_{l,i}) \tag{2.3}$$

$$m_{i,j}^1 = \frac{1}{n_i^1} \cdot \sum_{l=1}^{n} h_{l,i} \cdot t_{l,j} \tag{2.4}$$

$$m_{i,j}^0 = \frac{1}{n_i^0} \cdot \sum_{l=1}^{n} (1 - h_{l,i}) \cdot t_{l,j} \tag{2.5}$$

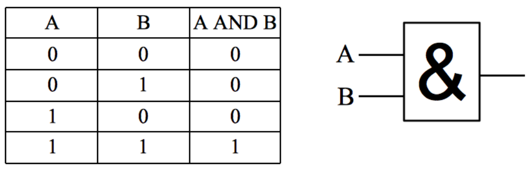
$$\mathbf{R} = \mathbf{mean}(\mathbf{m^1}) - \mathbf{mean}(\mathbf{m^0}) \tag{2.6}$$

2.2.4 Template Attacks

Template attacks are a type of DPA attacks. In all previous procedures it was assumed that the attacker has little knowledge about the power consumption of the device to be attacked. Template attacks are template-based DPA attacks that already have a collection of information about the power consumption in advance and use it during an attack to characterize the new consumption values. Clearly, this means that the new traces can be correctly classified using previously known information and thus the correct key can be captured. Templates are optimal for knowing the characteristics of the device and its consumption in advance.
The execution of the template attacks differs according to the calculation of the Hypothetical Intermediate Values. In the further course Bayes' theorem is applied to the learned values and the hypothetical values. With the result of this model a prediction with the test data is carried out. At the end, the probability of the key is determined, whether it yields the correct key. This process is performed for all possible keys.

3 Field Programmable Gate Array

A Field Programmable Gate Array (FPGA) is, as mentioned in the introduction, a logical device that can implement user-defined logic using logical functions [Ama18]. Logical algebra defines the following operations: AND, OR, NOT. The AND and OR operations are operations that generate an output with at least two inputs. The input and output can be displayed in a truth table.
A truth table is a table in which all inputs and outputs of an operation are written. The size of a truth table is 2^n, where n is the number of inputs. A certain type of truth table is the look-up table, which plays a big role in the function of an FPGA. In Figure 3.1 we see a small example of a truth table.



A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Figure 3.1: Truth table of an AND gate with two inputs

3.1 Construction Of A FPGA

The general structure of an FPGA consists of the following components:

- **Logic Element:** The logic element of an FPGA is a logical block in which the implementation can be realized. Logic blocks are used to build and execute the implementation. Any desired circuit can be implemented in this block. The logic element also serves as a memory function for the implementation.

  The following two methods are used in FPGAs:

  - **Lookup-Table**: A lookup-table (LUT) is a memory table whose input generally refers to the address of its output. The output is statically defined so that time-consuming calculations can be avoided in order to shorten runtimes. The output values of a look-up table are output values of a specific function. All possible inputs and outputs of the function are stored in the table. The look-up table eliminates the need to execute the function.

  - **Multiplexer**: A multiplexer (MUX) is a circuit that has a number of inputs and one output. Control signals, which are considered as additional inputs, determine which input is switched through to the output. Multiplexers can thus convert parallel inputs into serial outputs. An example for a MUX would be n-MUX. The circuit has n control signals with which 2^n inputs can be controlled. There is a counterpart to the MUX called demultiplexer. A demultiplexer (DEMUX) has one input and several outputs. In addition, the DEMUX, like the MUX, has additional control signals which are used to determine the through-connection. These signals switch the input to the specific output.

  Both of the methods use programmable logical blocks. Flip-flops are one example. A flip-flop is an electrical circuit that has two stable outputs. The state changes not only of current inputs, but also of events that have occurred before.

- **I/O Element:** The I/O element is a programmable external interface which is used for the input of external signals and for the output of internal signals. An external signal can be sent by different hardware, for example by a button or by a cable over a pin from another device. Internal outputs are used to transfer information from the FPGA to another device (such as a computer).

- **Connecting Element:** The connection element is used to connect blocks and elements. It is a block consisting of wiring channels, switch blocks and terminal blocks. The block can be a connection between logical blocks as well as a connection between a logical block and an I/O block.

3.2 Programming Technologies

FPGAs generally consist of various programming techniques that can be used to execute the implementation. The following techniques are widely used in FPGAs:

- **Flash Memory:** Flash memory is a non-volatile memory on which the implementation cannot be reconfigured. The program is implemented electrically and can only be read in the further course. For a configuration of the program, the program must be completely reinstalled.

- **SRAM Technology:** Compared to flash memory, SRAM Technology is a volatile static memory on which the program can be reconfigured at will. The disadvantage of this technology compared to flash memory is that the volatile memory is reset when the power is interrupted, while the flash memory retains everything.

4 AES-128 Implementation

As already mentioned in the introduction, the AES encryption algorithm has been implemented on the board for the power analysis attacks.

4.1 Advanced Encryption Standard

The Advanced Encryption Standard (AES) is a block cipher that performs 10, 12 or 14 rounds depending on the key size [Dae99]. The first round performs the following operations: `KeyExpansion`, `AddRoundKey`, `SubBytes`, `ShiftRows`, `MixColumns`. In every other round (except the last round) all operations except `KeyExpansion` are performed. In the last round `MixColumns` are omitted.

4.1.1 Operations

In the following, the operations of the implemented AES encryption algorithm are explained in more detail:

`KeyExpansion`: This operation generates all necessary round keys at the beginning of the encryption. The individual keys are generated depending on the previous round key. The following two functions are performed: `SubWord` and `RotWord`. `SubWord` takes a four-byte input word and applies the S-Box to each byte. `RotWord` takes a four-byte input word and performs a ring shift.

`AddRoundKey`: This operation is executed at the beginning and at the end of each encryption round. The bitwise block and the XOR round key are linked. This operation is the only one that makes the algorithm dependent on the key.

`SubBytes`: This operation applies the S-Box to each byte entered. The transformation is non-linear.

`ShiftRows`: This operation performs a shift operation on the 16 Bytes entered in columns. The lines are shifted cyclically. The first line is not moved, the second line is moved one position, the third one moves two positions and the fourth line moves three positions.

`MixColumns`: This operation processes the column-by-column bytes, column by column. The data is mixed within the columns by means of the Galois field operations.

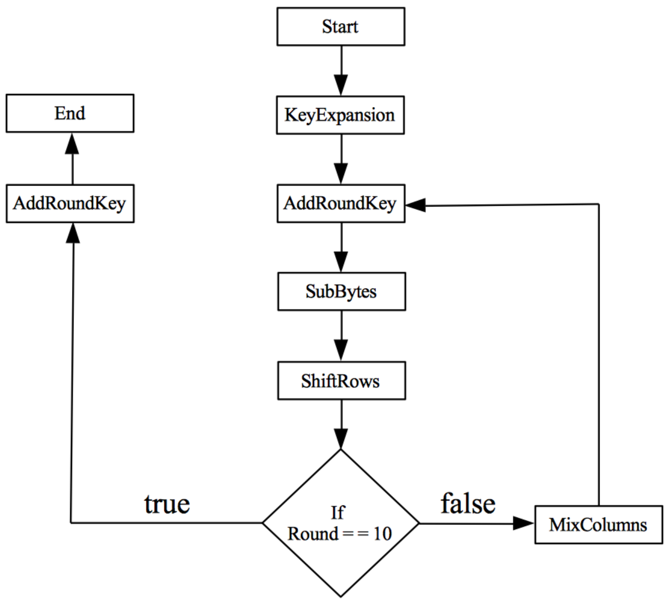


Figure 4.1: Execution of AES-128

4.1.2 Modifications On AES

In this work we used a Verilog AES 128-bit encryption implementation on the board. AES encryption has been slightly modified for the general operation of the AES. `KeyExpansion` is not performed in the first round in this implementation, but at the beginning of each round. Thus, not all round keys are generated at the beginning, but each round key in the respective round.
Another change is the S-Box. The S-Box was not implemented as usual as a table that takes one byte and omits the respective value from the table. This variant uses only the operations XOR and AND. The circuits are depth 16 and the two operations perform 128 calculations. This variant of the S-Box is thus more efficient than the usual variant [BP11].

4.2 Communication Interface

Universal Asynchronous Receiver Transmitter (UART) was implemented as communication interface for sending bytes from the FPGA to the computer and receiving bytes from the computer.

The UART transmitter can send a byte only bitwise. For this, it initially sets the line to a one. Once the start is given, the line is set to zero and every 1250 clock cycles one bit is sent over the line. After 8 bits are sent, the line is finally reset to 1 to give the receiver the ready signal. The UART receiver basically works much like the transmitter. The receiver waits for the line to be set to 0 by the transmitter and then starts reading. In the first step, he checks the start bit again after 625 clock cycles. Then he waits 1250 clock cycles and reads the first bit. Then he reads the remaining bits every 1250 clockcycles. At the end, he still waits for the transmitter to give the end signal to process the 8 bits read.
A Python code was written for sending and receiving the bytes. At the beginning the declared key is sent byte by byte to the FPGA. Then 16 bytes of plain text are generated at the beginning and also byte by byte to the board. The generated bytes are also written to a text file for later analysis. Now the program waits until the decryption is finished and the ciphertext comes back byte by byte. This process is performed for a number of plain texts. If now 5,000 plain texts have been sent, the program waits for the start of the next run. The next run is started by pressing a button. The reason for these runs is that the meter can only measure limited encodings during the analysis. Since the measurement had to be carried out several times, this method was the most effective.

4.3 Construction Of The Implementation

The code contains five modules: Top Module, AES Module, S-Box Module, UART Transmitter Module and UART Receiver Module. The top module is the main module and serves as a kind of state machine, which gives the other modules the start and end command. It receives 16 bytes from the UART receiver and sends them to the AES module. At the end of the encryption, it sends the encrypted 16 bytes to the UART transmitter.

4.4 Properties Of The Implementation

The AES on the board finishes in 450 cycles. An AES with better clock rate is possible, so that for example several Sbox modules could be built to work more in parallel. Unfortunately, this is not possible on the ICE40HX8K-CT256 because it has a low capacity (lookup-table limited) and thus only one Sbox has to work iteratively for all inputs. Although the capacity was limited, but the runtime increased slightly.
The board has a capacity of 7680 logical elements (lookup-table capacity). The implementation requires a capacity of 2616, of which 528 is used by the UART and 2088 by the AES implementation. Thus, the implementation has been reduced so that more than half of the capacity is still available on the board.

The implementation requires a total of 4956 cells. The AES itself requires a number of 3635 cells. Cells is the sum of the number of flip-flops, the size of the lookup-table and the carry.

4.4.1 Comparison To Other Implementation

Now let's look at two other AES implementations. In [MPL+11], one of the AES implementations goes through 160 clockcycles, while the other implementation requires 1032 clockcycles. The first AES requires few clock cycles because some of the operations are performed in parallel. Since our implementation was not constructed in parallel due to lack of capacity, a comparison of the two implementations is not practical. The comparison with the second implementation is possible because it runs through serially like our implementation and no operation is performed in parallel. Our implementation is the better choice when comparing the two, because despite the reduced capacity, our implementation requires less than half of the clock cycles.

5 Practical Execution

5.1 Measurement Setup

Several components are necessary for the execution of an attack. In the following these components are described with the characteristics and the function during the attack:

Oscilloscope: The Mixed Signal Oscilloscope MSO64 was used to record the traces. The device has 4 channels as input. These channels can measure in a bandwidth from 1 to 8 GHz. The maximum sample rate is 25 GS/s. The device has a USB interface, with which the measured data can be extracted. The traces are saved as file format Waveform (.wfm).

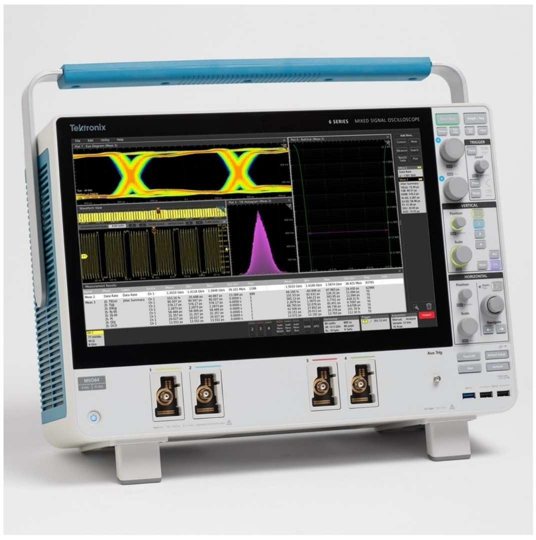


Figure 5.1: Oscilloscope MSO64 ([Tek])

Cryptographic Device: An FPGA is used to perform the side-channel attacks. For this, the SRAM-based ICE40HX8K-CT256 was chosen. The board has 8 LEDs, which the user can program. It also has a SPI Flash for programming the configuration. By means of a USB mini interface the communication to the computer is made, whereby this is used at the same time as power source for the board. In addition, the board has 0.1 inch holes for user ports. The board is inexpensive and well equipped for its price. It can also be programmed by means of open source software, which was also done in the practice of this work. The board has 7680 logic elements, so the board's capacity is quite limited compared to more expensive boards.

The cryptographic device contains as code the AES-128 encryption algorithm with UART as communication interface. This interface enables the computer to send plain text and keys bit by bit to the FPGA, which performs the encryption and returns the ciphertext bit by bit.

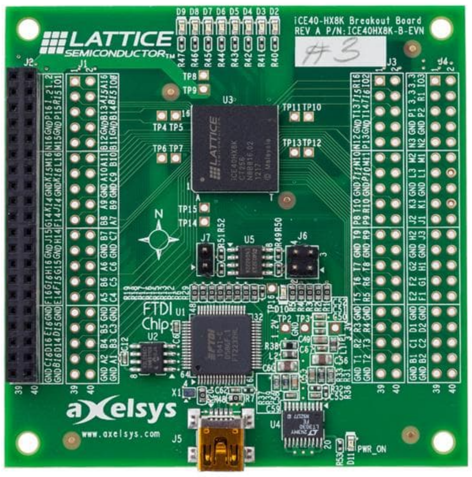


Figure 5.2: ICE-HX8K Board ([Lat])

EM Probe: An electromagnetic measurement was performed on a capacitor of the FPGA during encryption. A capacitor can statically store energy in an electric field. With the help of the EM sample an indirect measurement of the energy consumption at the capacitor can be carried out.

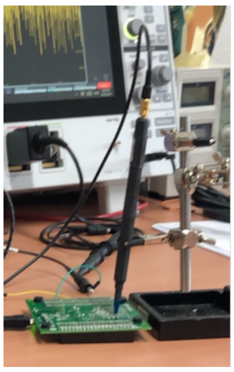


Figure 5.3: EM Probe

Computer: The computer is needed to send data to the cryptographic device and receive data. For communication, Python is used on the computer. In addition, the computer is also needed for the analysis of the measured data.

5.2 Measuring Side Channel Information

For the measurement, the ICE40-HX8K board was connected to the computer via a USB interface and the implementation was programmed on the board. The written Python code was used to connect the computer to the board with Serial in the dev folder (device: ttyUSB1). The connection was established with a baud rate of 9600.
The board was used in CRAM programming mode. This keeps the implementation on the board until the power connection is interrupted. The configuration will be cleared as soon as the power stops flowing. There is also a possibility to keep the configuration permanently on the board. The device has a SPI flash memory on which the configuration is stored. In this mode the configuration is always read from the flash memory.

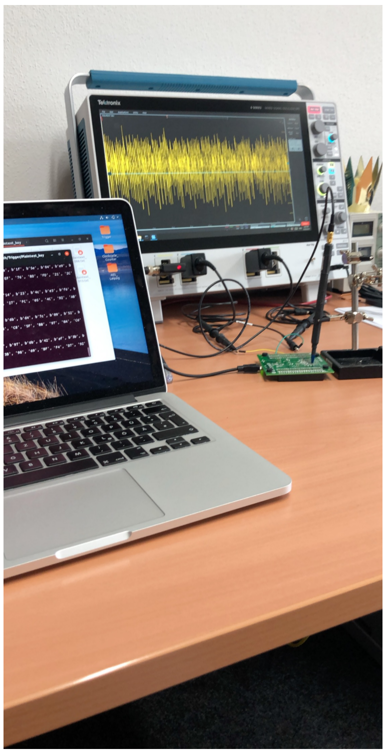


Figure 5.4: Measurement Construction

A trigger has been added to the implementation that tells the Oscilloscope when the AES starts. The 1-bit output register for the trigger is set to 1 before the start of the AES. The signal is left at 1 for 10 clock cycles and then reset to 0. Then the AES module starts. Pin R10 is used on the board for the trigger signal.

On the Oscilloscope only two input channels were necessary for the measurement. One channel was used for the input of the trigger signal. Therefore the channel was connected with a cable to the R10 pin and additionally to Ground (GND). The second channel was used for the electromagnetic measurement. Therefore the EM probe was connected to the second channel, which measured at the capacitor C43 of the microcontroller. The selection of this capacitor has the reason that there is a voltage of 1.2 Volt and an electrical capacitance of 1 uF (microfarad). The electrical capacitance was sufficient for the measurement. The microcontroller has further capacitors which supply a voltage of 1.2 Volt. These capacitors did not have enough electrical capacitance (0.1 uF) to perform the measurement.
After the experiment was set up, the configuration of the oscilloscope was carried out. An SL (sample rate) of 1.25 GS/s was set for recording. The first channel, the trigger channel, was activated and the trigger level adjusted so that the recording is started from the fall of the trigger signal. Then the second channel was configured so that only the AES is recorded. Only a sample size of 12500 was recorded, which corresponds to about two and a half rounds of the AES. The reason for this is that the key of the first round is the key searched for. In the first round of the AES, this key is added to the plain text and entered into the S-Box.
For the recording, 50,000 plain texts were randomly generated using the Python code and sent to the FPGA. In addition, these generated plain texts were also saved, since they are required for the analysis with the traces recorded afterwards.

5.3 Analysis And Results

The analysis was performed using the MATLAB program. The analysis required the ten trace files that were recorded and the ten plain text files that were encrypted. The traces were written into a matrix in Matlab in the correct order. In the same order, the plain texts were stored in a separate matrix. An additional script was required to read the traces, which can read data from a .wfm file. In addition, an AES S-Box was created as an array for the analysis.
During the analysis, the number of plain texts was reduced from 50,000 to 20,000, as the leak had already been detected and the analysis time was thus significantly reduced. In addition, the tracesize was reduced from 12,500 to 6,000. The reason for this is that the first round already takes place at less than 6,000 and only the first round is required for the analysis. Assume that the total number of traces is denoted by N, we denote the set of traces as follows:

$$T = \{t_i | t_i \text{ for } 1 \leq i \leq N \text{ is a side-channel trace corresponds to } p_i\}$$

The analysis or the attack was carried out with four different variants.

- Correlation-Based DPA (CPA) bit model was used as the power model.
- The second variant is also CPA, but was used here as power model hamming weight.
- Variant three is the Difference of Means.
- The fourth and last variant were template attacks.

Each of the variants performs the analysis for a single specific byte. This means that the analysis was performed for each byte of the key (i.e. 16 times). Each of the analysis variants calculates at the beginning the Hypothetical Intermediate Values of the searched key byte.

Algorithm 1: Calculation of Hypothetical Intermediate Values (c)

```
1 $p_1, \dots, p_n \leftarrow$ A set of plaintexts with $n$ element
2 $t \leftarrow$ Select a key byte to target                // $t \in \{1, \dots, 16\}$
3 for $0 \leq k \leq 255$ do
4     for $1 \leq i \leq n$ do
5         $c_{i,k} \leftarrow \text{sbox}(p_i(t) \oplus k)$;   // $p_i(t)$ represents the $t^{th}$ byte of $p_i$.
6 return $c$;
```

The Hypothetical Intermediate Values are calculated as follows: As can be seen in algorithm 1, all existing plaintexts whose traces were collected are first stored in a matrix p. In addition, the searched byte must be stored as variable t. Now xor is performed for each key k and for each row of p_i. The reason for the column selection is that the column index of p corresponds to the position index of the key byte. Finally, the results are given in the S-Box created before the analysis. The results are then stored in c and produce the Hypothetical Intermediate Values.

5.3.1 CPA Using Bit Model

The bit model was used in the analysis variant. As can be seen in algorithm 2, the first bit, the last significant bit (LSB), of the previously calculated Hypothetical Intermediate Values, i.e. of the S-Box output, is stored as matrix H. The resulting H matrix, also called Hypothetical Power Consumption, is checked for correlation with the traces T and inserted into the R correlation matrix. Then the matrix R is searched for the absolute maximum, which after analysis results in the index of the correct key byte in R.
In Correlation-Based DPA, most of the results were successful. With the exception of three bytes (byte 1, byte 3, byte 16), all parts of the key were successfully uncovered.

Algorithm 2: Calculation of CPA using bit model

```
1 $H \leftarrow \mathrm{LSB}(c)$
2 $R \leftarrow corr(H, T)$;            // $corr$ represents the pearsons correlation
3 $ind \leftarrow find(|R|) == (max(max(|R|)))$
4 **return** $ind$;
```

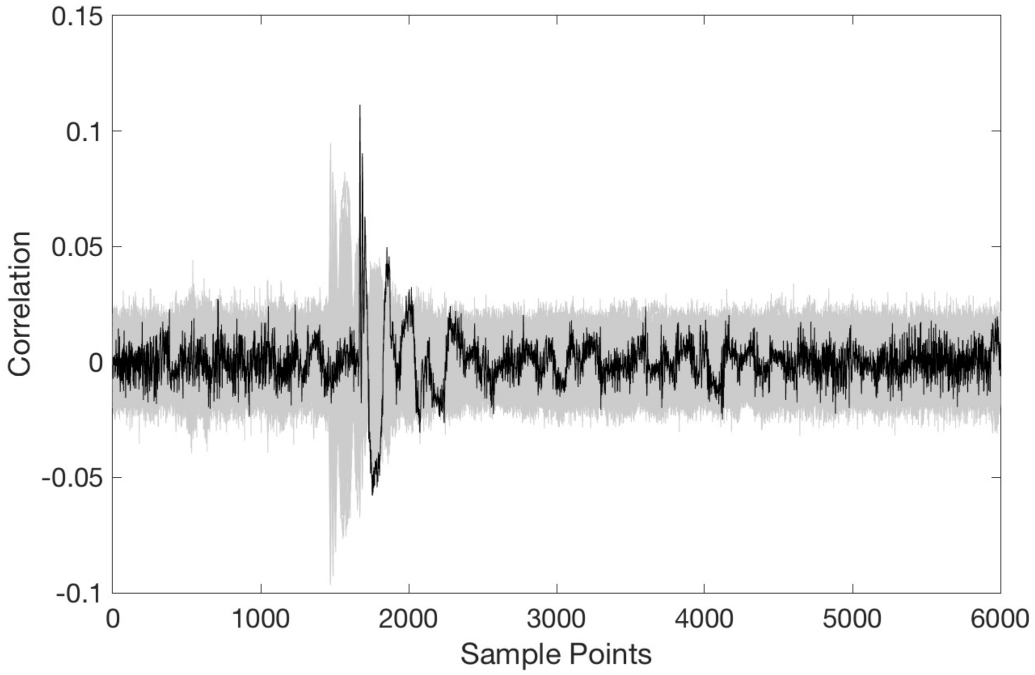


Figure 5.5: CPA using bit model result of byte 2

Let's take a closer look at the result of the second byte in Figure 5.5: Here we found the correct key, the key candidate no `7E`. In the figure, the grey peaks represent the correlation of all key hypotheses. The black peaks represent the correlation of the correct key in all sample points. Key `7E` has the highest correlation in sample point 1671 and is the correct key searched for according to the procedure.

The figure also shows further peaks at other sample points. This is because the columns of the Hypothetical Intermediate Values are not all independent of each other. This clearly means that if a column has a high correlation, higher correlations can also occur for other columns if these are dependent on the column.

The high correlation also depends on the number of traces used. Figure 5.6 shows the correlation of all keys of byte 2 from 1.000 to 30.000 traces in thousand steps. The key `7E` is marked black in the graphic. All other keys are shown in gray. You can see that the

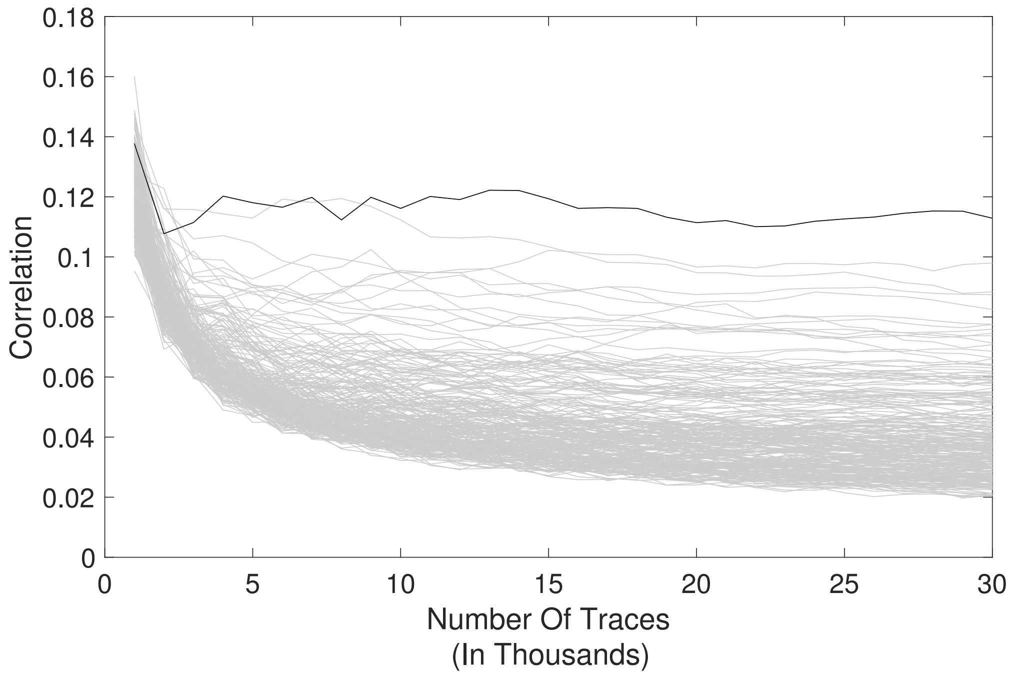


Figure 5.6: Number of Traces versus Correlation using bit model result of byte 2

correlation of key 7E with more traces is higher than all other keys. When performing the analysis with up to approximately 4,000 traces, the correct key could not be uncovered with byte 2. It can also be seen that the key would be discovered successfully with 5,000 traces, but there would be no leak with approx. 6,000 traces. Therefore, the number of traces for the leak plays an important role. As of 9,000 traces, the leak would always exist.

Now we look at another key byte we were looking for, the correct key of which could not be uncovered. The Figure 5.7 shows us the correlations of the searched key byte 1. The analysis of the byte showed the key 108 as the result of the absolute maximum of the correlation matrix. The correct key of byte 1 is key 2B. In the figure you can see that the highest peak in sample point is 1261, which was assigned to key 6C. However, the correct key in the graph also has a higher peak than all other keys. The peak in sample point 1463, i.e. the largest right peak in the graph, is the correlation of the correct key. The probable reason for the incorrect detection of the key is that another operation took place shortly before the first S-Box operation and the traces are probably not independent of the previous operation.

In Figure 5.8 we can look again at the correlation of key 2B, which is marked in black. It can be seen that the key 2B to 30,000 Number of traces cannot be uncovered. The correlation of the key will increase with the increase of the number, but unfortunately does not

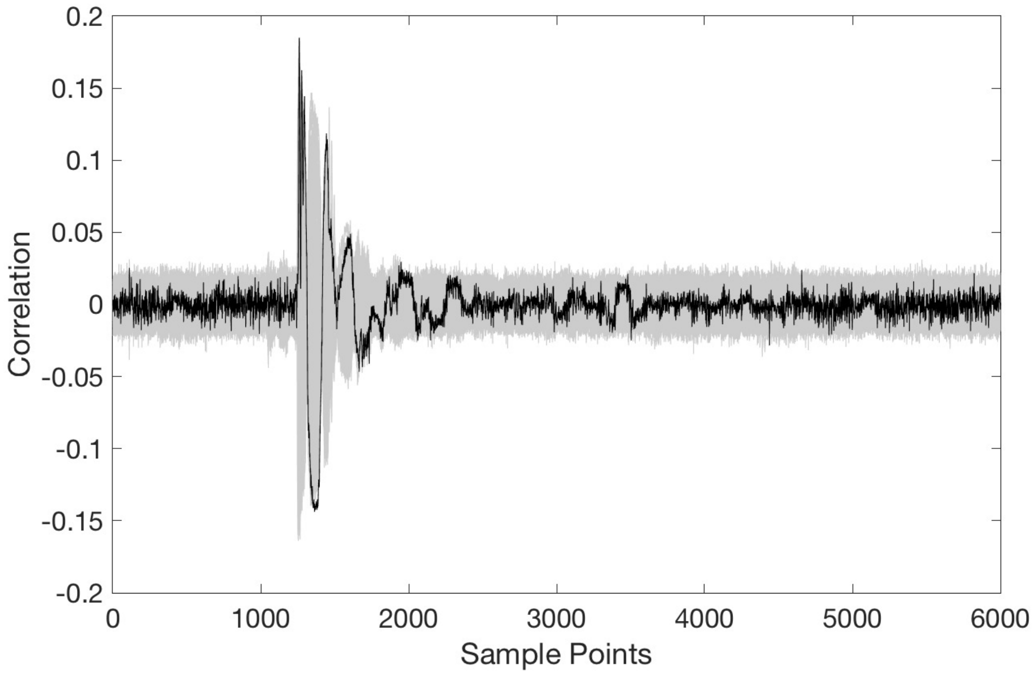


Figure 5.7: CPA using bit model of byte 1

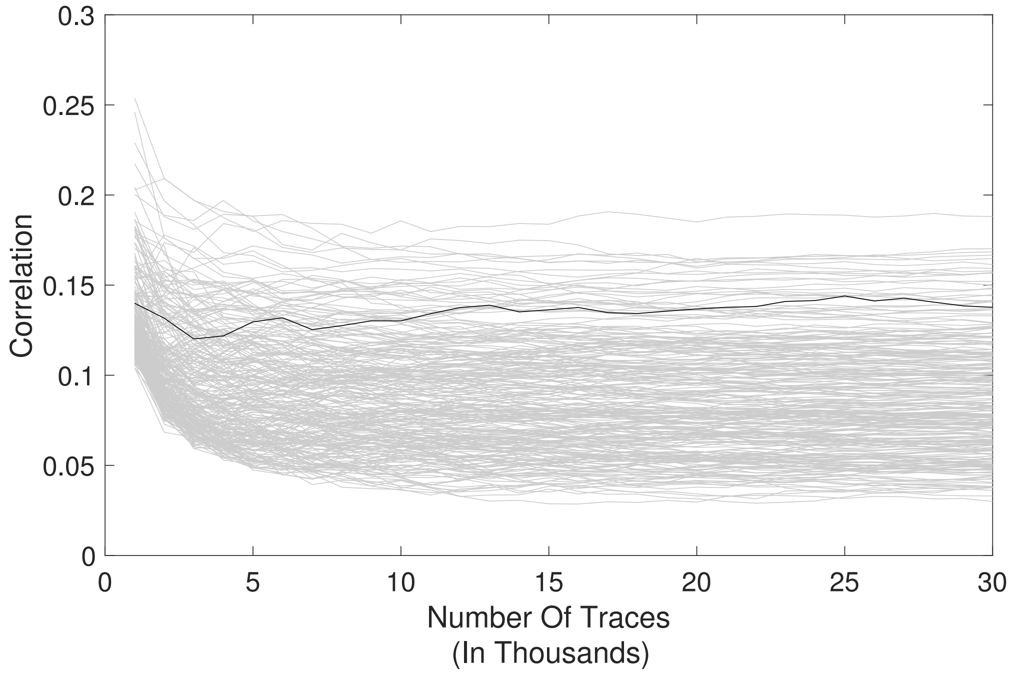


Figure 5.8: Number of Traces versus Correlation using bit model result of byte 1

exceed any other key.

5.3.2 CPA Using Hamming Weight

The analysis variant is the same variant as the CPA described above, but uses the Hamming weight as the power model. As in algorithm 3, not the first bit of the S-Box output is used, but all 8 output bits. The Hamming weight is calculated from all bits. For this the output matrices of all bits are added together, so that the matrix H results in the sum of the ones at each position. Then, as with the CPA using bit model, the matrix H is checked for correlation with the traces T and the resulting correlation matrix R is searched for the absolute maximum, which then results in the index of the correct key in R.

Algorithm 3: Calculation of CPA using Hamming weight

```
1 $H \leftarrow zeros()$
2 for $1 \leq i \leq 8$ do
3     $H \leftarrow H + c_i$;                  // $c_i$ represents the $i^{th}$ bit of $c_i$
4 $R \leftarrow corr(H,T)$;          // $corr$ represents the pearsons correlation
5 $ind \leftarrow find(|R|) == (max(max(|R|)))$
6 return $ind$;
```

The Correlation-Based DPA with the use of the Hamming weights as power model all keys exclusively of key byte 16 were uncovered.
Let's take a closer look at the result of the first byte in Figure 5.9: The correct key, key `2B`, was found successfully. If we take a closer look at the black peaks in the graph, we can see that several large peaks are visible. However, the correlation of the correct key is quite high and clearly recognizable. If we take a closer look at the Number of Traces versus Correlation in Figure 5.10, we can see that even with a smaller number of traces, the key can be uncovered. From about 6.000 number of traces a safe leak can be detected.
With the CPA using Hamming weight a high correlation of all bytes could be discovered, excluding key bytes 1 and 16. As can be seen in 5.11, key byte 5 has a leak from 1,000 Number of Traces. The graphic looks similar for all other bits. From this it follows that the method was quite efficient for the expert.

5.3.3 Difference Of Means

The Difference of Means method uses the bit model. The Hypothetical Intermediate Values are stored as matrix H. For each key and thus for each column of H (Hypothetical Power Consumption) two sets $\mathcal{U}$ and $\mathcal{V}$ are declared, which are empty at the beginning. The respective column is now checked for a 1 in all rows. If the case occurs, the same row

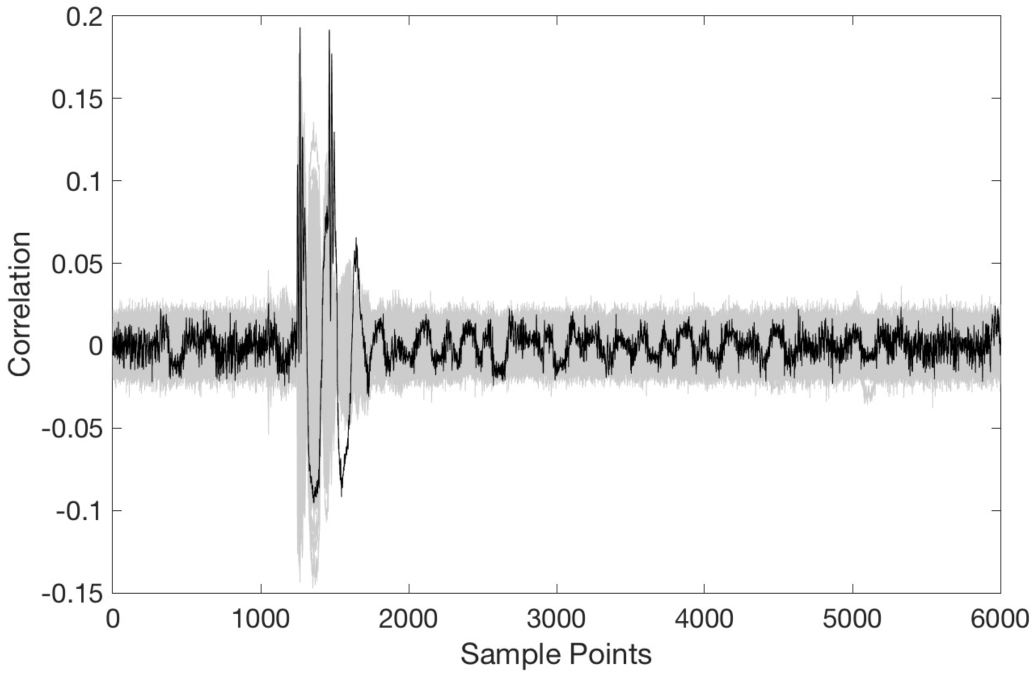


Figure 5.9: CPA using Hamming Weight result of byte 1

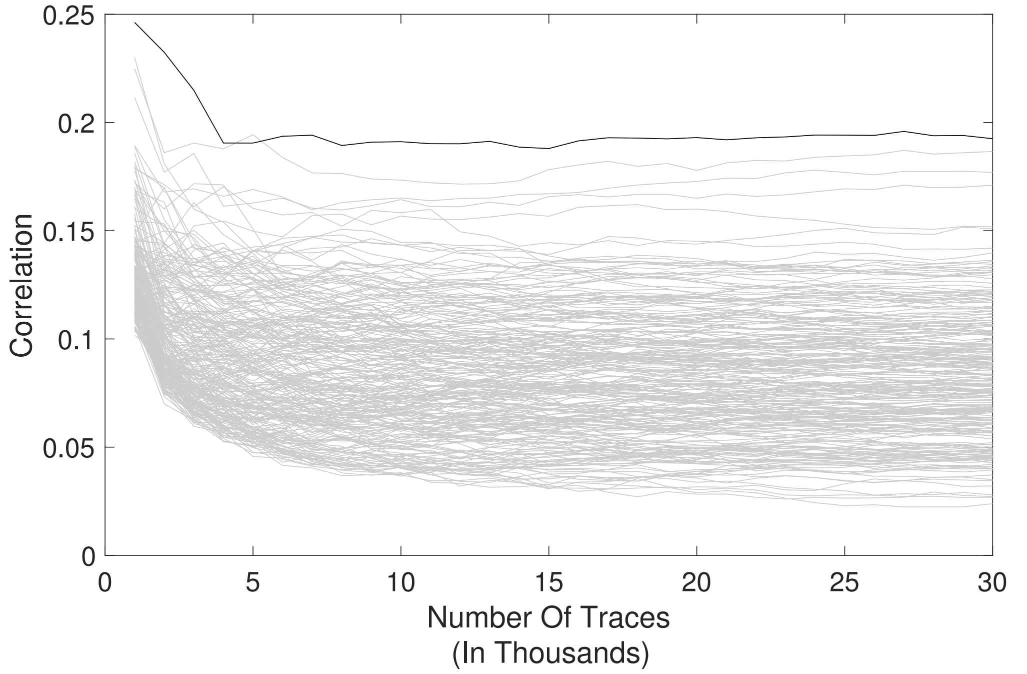


Figure 5.10: Number of Traces versus Correlation using Hamming Weight result of byte 1

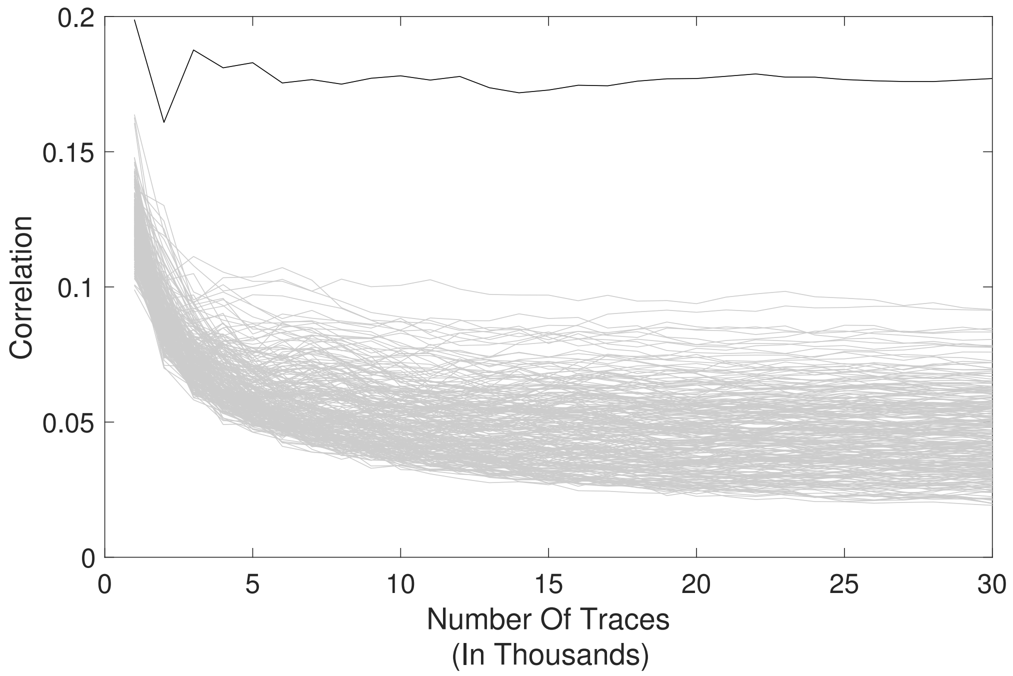


Figure 5.11: Number of Traces versus Correlation using Hamming Weight result of byte 5

of the matrix traces is stored as column of the matrix $\mathcal{U}$. If there is no 1, the same row of traces is stored in $\mathcal{V}$ as column. After all lines of traces have been stored in the correct set, the Means of both sets are calculated. Then the Mean of $\mathcal{V}$ is subtracted from the Mean of $\mathcal{U}$. The result is then stored line by line in the matrix R. The procedure is thus carried out for all possible keys. Then the matrix R is searched for the absolute maximum, which results in the index of the correct key in R.

Using the Difference of Means method, many of the keys were successfully determined. Of all 16 bytes, 4 bytes (byte 1, byte 3, byte 5, byte 16) could not be detected.

Now let's take a closer look at the result of the fourth byte in Figure 5.12: The correct key, key 16, was successfully determined. The black peak is the largest peak according to the graph and therefore the correct key according to the method. In the figure you can see a high peak, which is drawn in grey. This peak represents another key that could also be a key candidate after the calculation of the Difference of Means. From this one would come to the conclusion that the key shown in grey must be dependent on the correctly uncovered key.

Figure 5.13 shows us the result of the fifth key byte. We know that the largest black peak is not the correct key. The correct key is the grey peak. It can be seen that the black peak is represented negatively. Negative peaks are also considered, since the method is used to

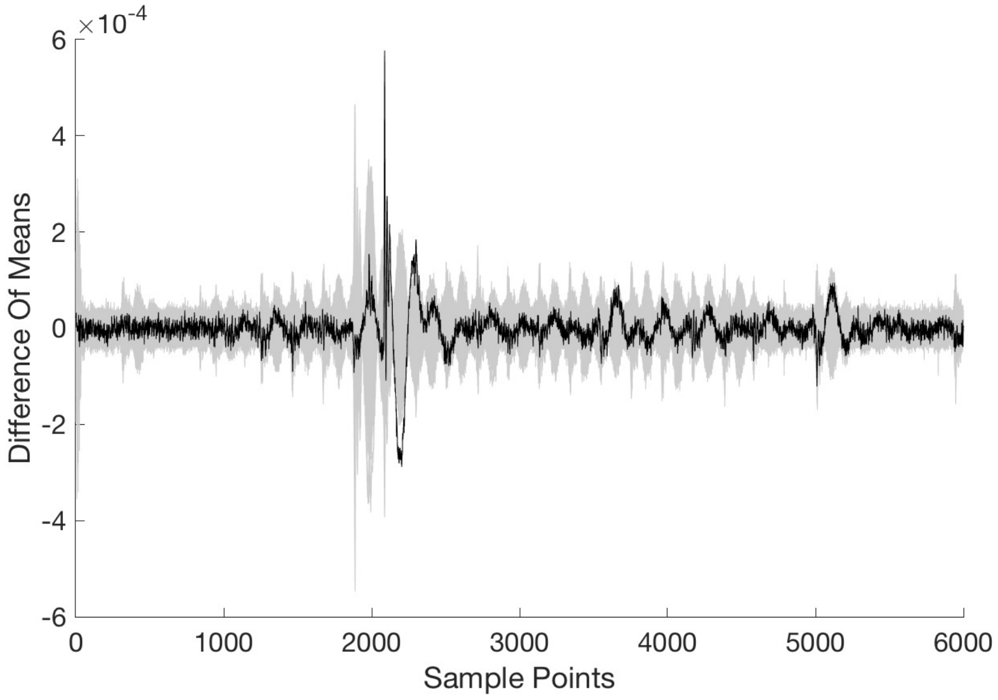


Figure 5.12: DPA result of byte 4

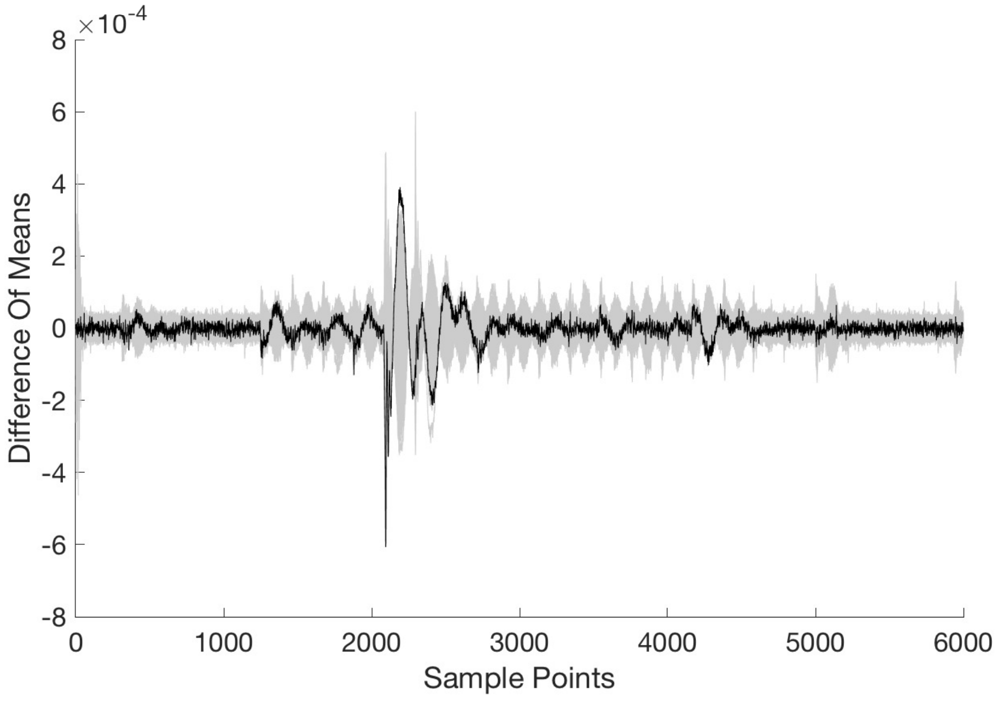


Figure 5.13: DPA result of byte 5

Algorithm 4: Calculation of Difference of Means

```
1  $H \leftarrow$ LSB($c$)
2  $N \leftarrow$ Number of all traces
3  for $0 \leq j \leq 255$ do
4      $\mathcal{U} \leftarrow \emptyset$ $\mathcal{V} \leftarrow \emptyset$
5      $u \leftarrow 1$ $v \leftarrow 1$
6      for $1 \leq i \leq N$ do
7          if $H_{i,j} == 1$ then
8              $\mathcal{U}_u \leftarrow t_i$
9              $u \leftarrow u + 1$
10         if $H_{i,j} == 0$ then
11             $\mathcal{V}_v \leftarrow t_i$
12             $v \leftarrow v + 1$
13     $R_j \leftarrow mean(\mathcal{U}') - mean(\mathcal{V}')$
14 $ind \leftarrow find(|R| == (max(max(|R|))))$
15 return $ind$;
```

search for the absolute maximum, which means that signs do not play a role. In addition, we can see that the grey peak, considered as an absolute result, is just as large as the black peak. In this case, however, the procedure was not sufficient to determine the correct peak.

5.3.4 Template Attack

The Gaussian Bayes model was used as template attack. In contrast to the other analysis methods, 30,000 plain texts and traces were used. The reason for this is that more traces were needed for the analysis to teach in and additionally for testing. 22,000 traces were used for learning. The remaining 8,000 were then used for testing.

The bit model is used for this procedure. For this purpose, the Hypothetical Intermediate Values are initially stored as Matrix L. The following calculations are performed for all possible keys: First the training data of the traces are stored in T^0 and the Hypothetical Intermediate Values in L^0. Then the test data of the traces are stored in T^1 and the Hypothetical Intermediate Values in L^1. Now a model of the two training matrices is created. The naive Gaussian Bayes is used as model here. After we have created the model of both data, we create a prediction of $model$ on the test data T^1 using predict. The label of this prediction is stored as matrix l. Now we check the test data L^1 for equality with the matrix label. The counter cnt counts now how many places of the comparison apply. Now we calculate the accuration rating by dividing the counter by the number of test data (8,000 data). For the key j the rating is stored in r_j. Finally, we search the vector results for the absolute maximum, which then gives the index of the correct key in r.

Algorithm 5: Calculation of Template Attack

```
$N \leftarrow$ Number of all traces
$N_t \leftarrow$ Number of traces for train
$r \leftarrow \emptyset$
$L \leftarrow$ LSB($c$)
for $0 \leq j \leq 255$ do
    for $1 \leq i \leq N_t$ do
        $T_i^0 \leftarrow t_i$
        $L_i^0 \leftarrow L_{i,j}$
    for $N_t + 1 \leq i \leq N$ do
        $T_i^1 \leftarrow t_i$
        $L_i^1 \leftarrow L_{i,j}$
    $model \leftarrow fitcnb(T^0, L^0)$;            // $fitcnb$ represents naive Bayes
    $l \leftarrow predict(model, T^1)$;      // $predict$ represents Prediction function
    $cnt \leftarrow 0$
    for $1 \leq i \leq N - N_t$ do
        if $l_i == L_i^1$ then
            $cnt \leftarrow cnt + 1$
    $r_j \leftarrow cnt/(ntraces - ntrain)$
$ind \leftarrow find(|r|) == (max(max(|r|)))$
return $ind$;
```

By means of template attacks many keys could be uncovered. With five of the keys (byte 1, byte 5, byte 7, byte 15, byte 16) the detection was not successful.
The Figure 5.14 is the graphic of the ninth key byte. Compared to all other methods we have a completely different picture here. Now we no longer see the keys using the sample points, but the key hypothesis and its accuration rating. This clearly means that we have an overview of all keys and the probabilities on the possible key. The key hypothesis 171 has the highest peak and is therefore the right key we are looking for. The peak is clearly distinguishable from the others in the graph.
If we now look at Figure 5.15 of key byte 15, we can see that it contains several high peaks. In this example, the correct key 79 does not have a high peak. The reason for this is that probably more training data and also more test data is needed.

5.3.5 Comparison of the Attacks

A comparison of all the attacks carried out showed that Correlation-Based DPA with Hamming weight as the power model revealed most of the bits. While 13 correct key bytes

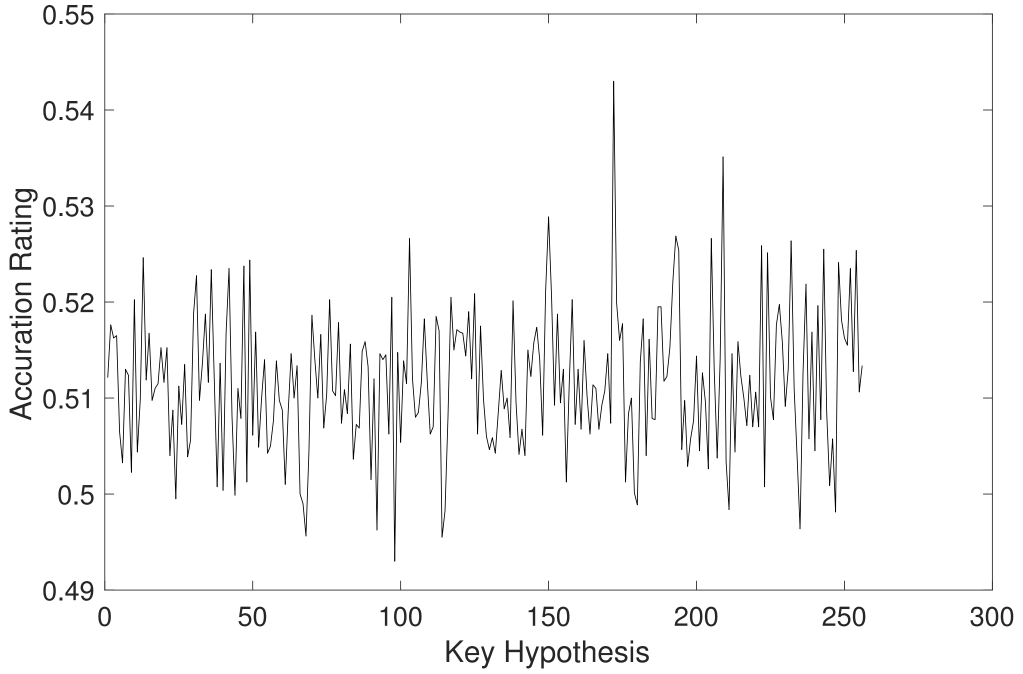


Figure 5.14: Template Attack result of byte 9

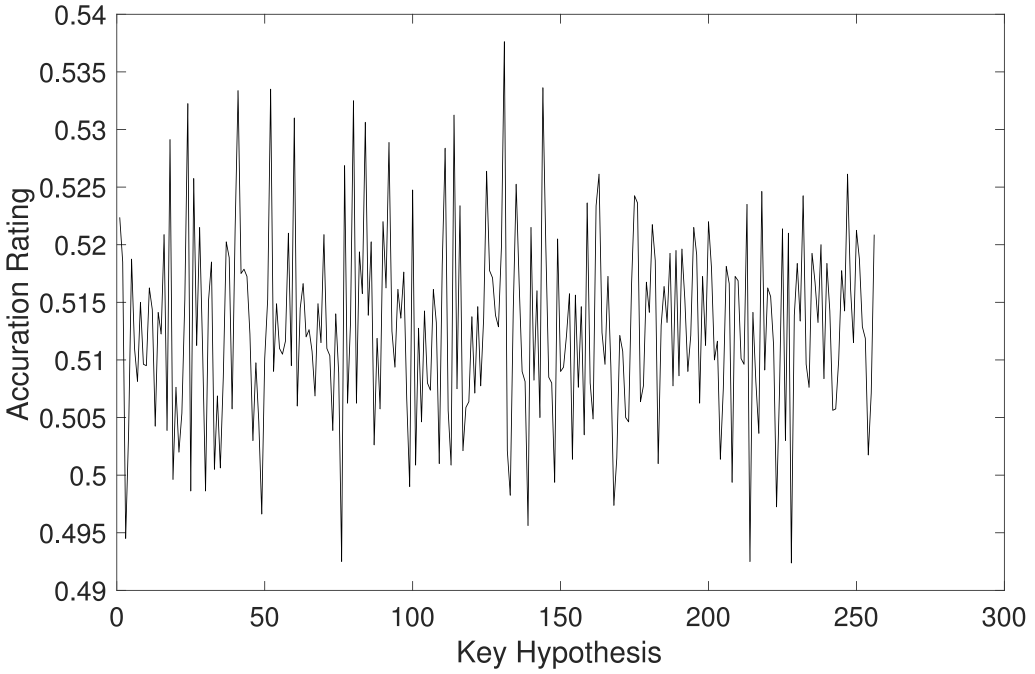


Figure 5.15: Template Attack result of byte 15

were detected with the bit model, 15 correct keys were found with Hamming weight. Let's look at the Figure 5.16 and Figure 5.17 of the sixth key byte for comparison. In both methods, the key was correctly uncovered. The figures clearly show that the peak at Hamming weight is much higher than the peak of the bit model. Basically this means that the correlation at Hamming weight is much higher than the correlation at bit model. When comparing both methods, the Correlation-Based DPA is clearly the better choice for this experiment.

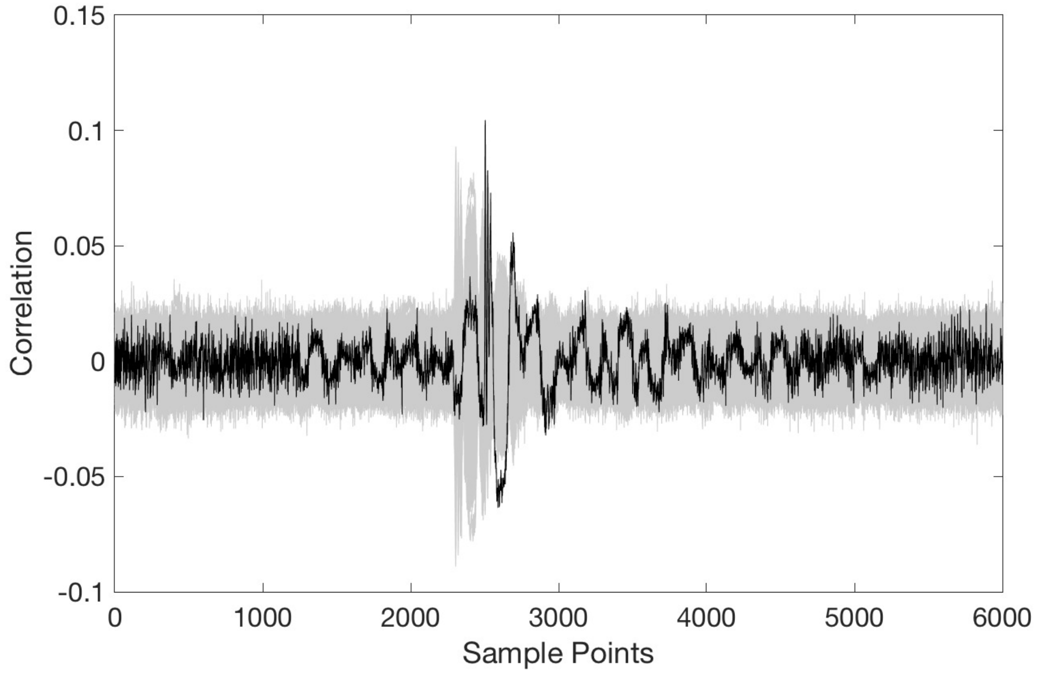


Figure 5.16: CPA using bit model result of byte 6

If we take a look at the Figure 5.18 of the sixth key byte from the DPA, the difference between the graphics of the CPA and the DPA is clearly noticeable. The peaks of the DPA are much cleaner and clearer. Likewise, all other keys (grey peaks) are much clearer and better readable than the Correlation-Based DPA.

In comparison to all methods, the template attacks provide the clearly better readable graphics. Let's look at Figure 5.19 from the sixth key byte. While the absolute maximum must be determined for all analyses in order to determine the index of the key, the key can be taken directly from the graphics of the template attacks. In contrast to all other methods, the mapping of each individual key next to each other is clearly advantageous for the reader.

Key byte 16 could not be successfully determined by any of the methods. By increasing the

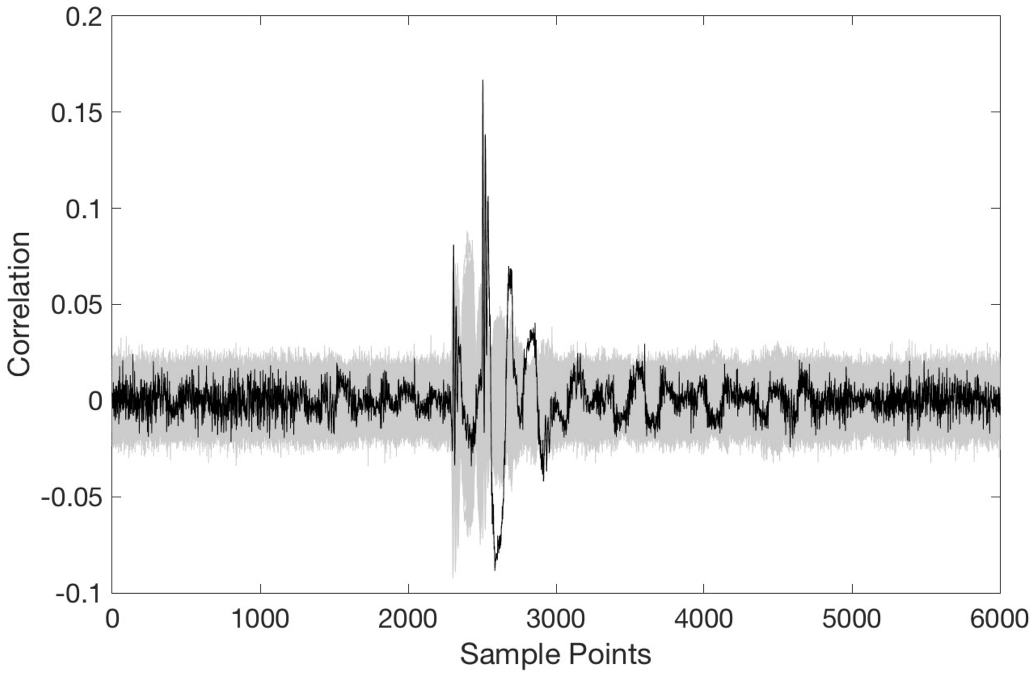


Figure 5.17: CPA using Hamming weight result of byte 6

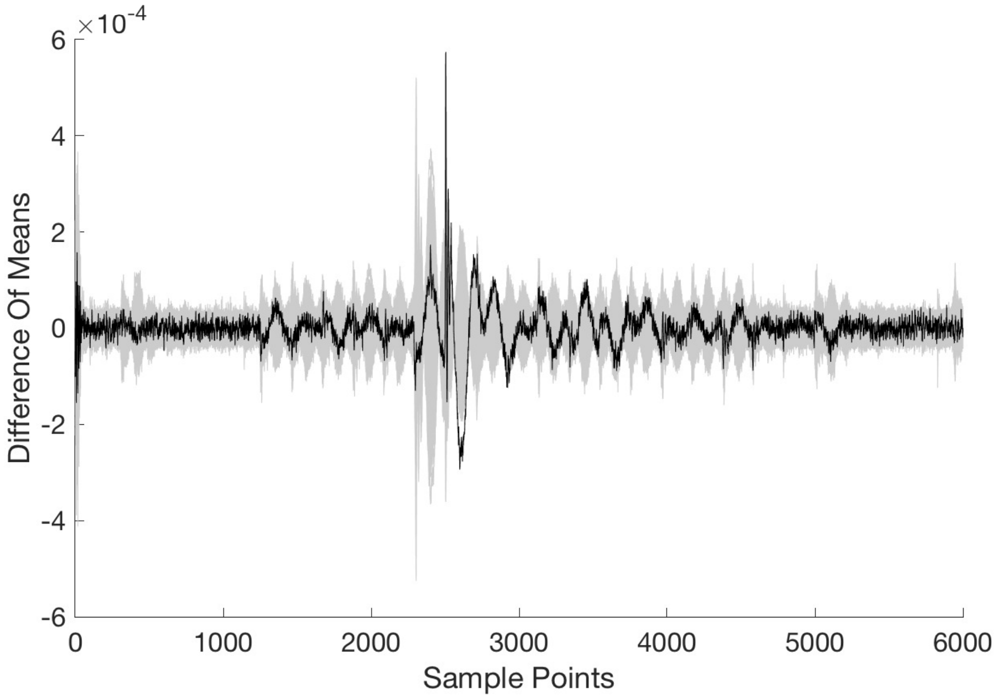


Figure 5.18: DPA result of byte 6

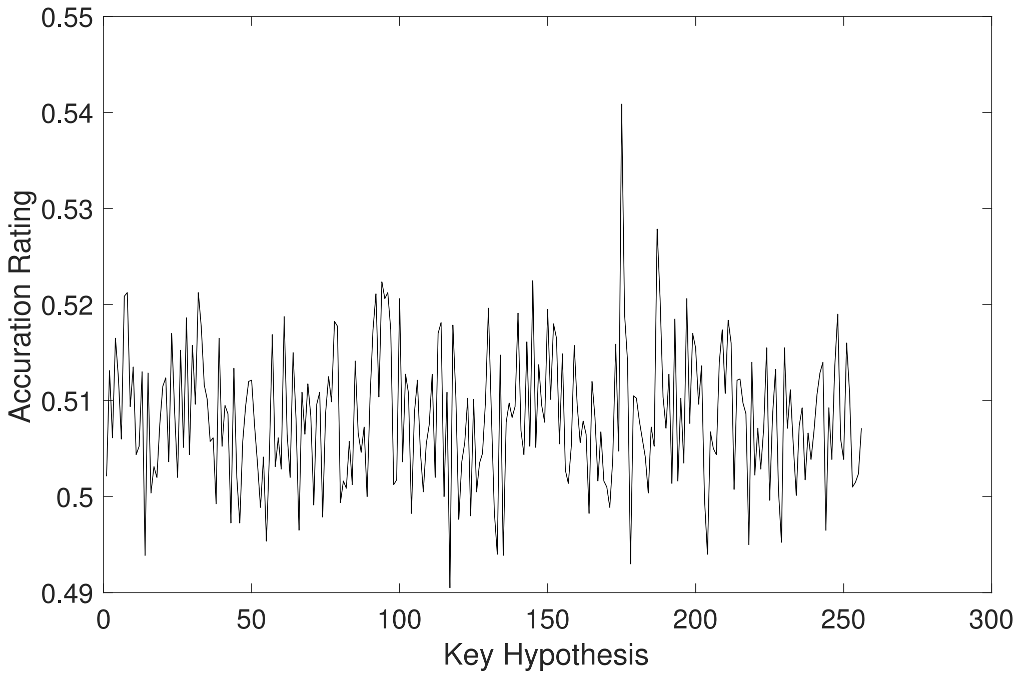


Figure 5.19: Template Attack result of byte 6

number of plain texts and traces, no result would probably be obtained, since the reason does not depend on the procedures themselves. The probable reason for this is that during and after the sixth S-Box operation another operation manipulates the measurement in phase.

In many cases, the procedures were able to deliver the correct results. Ten of the six correct keys could be determined correctly in all analyses. The Correlation-Based DPA was the only one able to provide the correct result for one key byte. All other keys were able to correctly detect at least two of the methods.

With the most efficient runtime and the most correct results, Correlation-Based DPA is the best choice for the whole experiment. The most inefficient choice is Difference of Means variant. The analysis had the longest runtime and provided almost no information.

6 Conclusion And Outlook

Finally, we programmed a Verilog implementation of the AES encryption algorithm on an FPGA. We have seen that the implementation can either be fast, but requires more capacity, or can have a longer runtime with less space requirement. The implementation has been designed to be as small as possible so that it fits even on low-capacity FPGAs. To make this possible, the implementation had to be built so that the whole algorithm would run serially. Also, the parallel run program would have been easy to construct. Due to the lack of capacity of our FPGA, this variant was difficult to implement. The component with the largest space requirement was the S-Box. For a parallel run you would have to create 16 S-Box modules to process all bytes simultaneously. In our variant, we constructed an S-Box that accepts all 16 inputs one by one.
Now that we have built an implementation and implemented it on the FPGA, we were able to determine and record the energy consumption by indirect measurements using the electromagnetic field of the FPGA during the encryption phase.
Now we have carried out different analysis methods with the recordings and the plain texts as input. The template based attack could deliver the fewest keys. The reason we suspected for this was that the information, i.e. the templates, was not sufficient to obtain the key for new entries. With a higher number of traces than the template, the procedure is probably the strongest variant of DPA attacks. The calculation of the template attack took some time. The runtime of the procedure is not necessarily efficient.
The Difference of Means variant could not detect three of the keys. The procedure provides clear results. If we take a look at the graph in the previous chapter, we can see that the method could provide the cleanest and clearest graph. The peaks are much clearer. With the least efficient runtime, the Difference of Means method, despite its clear results, would be our last choice as a method of analysis.
Correlation-Based DPA was the best choice as a method in our experiment to uncover the key. When choosing the bit model as power model all but three keys could be uncovered. After changing the power model and using the Hamming weight model, we were able to uncover all keys except the last key. This result was the best achieved result in our implementation. The runtime of the Correlation-Based DPA has the most efficient of all procedures. The execution was carried out in seconds and the results were delivered directly. With efficient runtime and the best possible solution, the procedure was the best method for our experiment.

At the end of all the experiments, we concluded that there is still a large security gap for attacks of this kind. Instead of focusing only on the mathematical security of the cryptographic algorithms, attacks of this kind must also be the focus of security.

References

[Ama18] H. Amano. *Principles and Structures of FPGAs*. Springer, 2018.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 16–29, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BP11] Joan Boyar and René Peralta. A depth-16 circuit for the aes s-box. volume 2011, page 332, 01 2011.

[Buc16] J. Buchmann. *Einführung in die Kryptographie*. Springer-Lehrbuch. Springer Berlin Heidelberg, 2016.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, pages 252–263, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 13–28, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Dae99] Joan Daemen. Aes proposal : Rijndael. 1999.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems CHES 2001*, pages 251–261. Springer, 2001.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *Advances in Cryptology–CRYPTO 2014*, pages 444–461. Springer Berlin Heidelberg, 2014.

[KJJ98] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Introduction to differential power analysis and related attacks. 1998.

[KJJR11] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1(1):5–27, 2011.

[Lat] Lattice semiconductor. `http://www.latticesemi.com/en/Products/DevelopmentBoardsAndKits/iCE40HX8KBreakoutBoard.aspx`.

[LP05] Kerstin Lemke and Christof Paar. *Physical Attacks*, pages 458–459. Springer US, Boston, MA, 2005.

[Man03] Stefan Mangard. A simple power-analysis (spa) attack on implementations of the aes key expansion. In *Proceedings of the 5th International Conference on Information Security and Cryptology*, ICISC'02, pages 343–358, Berlin, Heidelberg, 2003. Springer-Verlag.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag, Berlin, Heidelberg, 2007.

[MPL$^+$11] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of aes. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 69–88, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[Nov02] Roman Novak. Spa-based adaptive chosen-ciphertext attack on rsa implementation. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography*, pages 252–262, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[Tek] Tektronix. `https://de.tek.com/oscilloscope/6-series-mso-mixed-signal-oscilloscope`.