Masterarbeit

Evaluierung eines NFC-Tags welches AES und ECDSA unterstützt

Thomas Korak

t.korak@student.tugraz.at

Institute for Applied Information
Processing and Communications (IAIK)
Technische Universität Graz
Inffeldgasse 16a,
8010 Graz, Österreich

Begutachter: Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Karl-Christian Posch
Betreuer: Dipl.-Ing. Thomas Plos

Graz, im September 2011

Master's Thesis

Evaluation of a Security-enabled NFC Tag with AES and ECDSA

Thomas Korak

t.korak@student.tugraz.at

Institute for Applied Information
Processing and Communications (IAIK)
Graz University of Technology
Inffeldgasse 16a,
8010 Graz, Austria

Graz University of Technology

Assessor: Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Karl-Christian Posch
Supervisor: Dipl.-Ing. Thomas Plos

Graz, September 2011

Abstract

Near-Field Communication (NFC) is a contactless communication technique based on Radio-Frequency Identification (RFID) technology. Tags that are used in NFC systems are limited in power consumption and chip size. These limitations make integration of strong cryptographic security to such tags challenging. It is not enough to use a secure algorithm but also the implementation has to be secure against attacks like Side-Channel Analysis (SCA). In this work we evaluate an NFC prototype tag with advanced security features concerning its resistance against SCA attacks. The cryptographic algorithms implemented on the tag are the Advanced Encryption Standard (AES) and the Elliptic Curve Digital Signature Algorithm (ECDSA). In order to achieve resistance against this kind of attack, the NFC tag has two countermeasures integrated: shuffling and the insertion of dummy rounds. The main findings are that the countermeasures significantly increase the effort to reveal the secret key of the tag. The attacks in this work are performed under laboratory conditions with detailed knowledge of the device. For an attacker without this detailed knowledge, the effort for succeeding the attack will be much higher. Furthermore, we evaluate techniques to decrease the impact of the countermeasures on the attack complexity. One of these techniques is the detection of dummy rounds. The presented detection algorithm allows to reduce the attack complexity by a factor of 16. Besides evaluation of the SCA resistance of the prototype extensive functionality tests are performed to guarantee that the chip works according to the specifications.
With the SCA results on the cryptographic primitive we underline that it is not enough to use a secure algorithm but also the implementation has to be resistant against this kind of attacks. The usage of countermeasures can increase the complexity of power-analysis attacks significantly but the implementation has to be done carefully. It must not be possible for an attacker to reveal information about the parameters like the number of inserted dummy rounds used by the countermeasures.

Keywords: NFC, RFID, Side-channel attacks, AES, ECDSA, DPA

Kurzfassung

Nahfeldkommunikation (NFC) ist eine kontaktlose Kommunikationstechnik, die auf Radiofrequenz- Identifikation (RFID) basiert. Transponder, welche in NFC-Systemen eingesetzt werden sind sowohl in der Leistungsaufnahme sowie in der Chipgröße limitiert. Dadurch ist es schwierig, sichere kryptografische Algorithmen auf diesen Transpondern zu implementieren. Des Weiteren ist es nicht ausreichend, einen sicheren Algorithmus zu verwenden, auch die Implementierung des Algorithmus muss sicher gegen Attacken sein. In der folgenden Arbeit wird ein NFC-Transponder evaluiert, auf welchem kryptografische Algorithmen implementiert sind. Zur Evaluierung der Sicherheit werden Seitenkanalattacken (SCA) eingesetzt. Auf dem Transponder sind zwei Algorithmen implementiert, der Advanced Encryption Standard (AES) sowie der Elliptic Curve Digital Signature Algorithm (ECDSA). Um Seitenkanalattacken zu erschweren, sind auf dem Transponder zwei Gegenmaßnahmen integriert: Das Einfügen von so genannten Dummyrunden und das Randomisieren von Operationen (Shuffling). Das Ergebnis der Attacken ist, dass die Gegenmaßnahmen den Aufwand für das Finden des verwendeten Schlüssels signifikant erhöhen. Unsere Attacken wurden unter Laborbedingungen durchgeführt und es stand detailliertes Wissen über den attackierten Chip zur Verfügung. Für einen Angreifer, welcher dieses Wissen nicht hat, ist der Aufwand für eine erfolgreiche Attacke noch wesentlich höher. Es wurden auch Methoden evaluiert, welche den Einfluss der Gegenmaßnahmen auf die Komplexität der Attacke vermindern. Das herausfiltern der Dummyrunden ist eine der Methoden, welche eingesetzt wurden. Mit dem verwendete Algorithmus zum Erkennen der Dummyrunden konnte der Aufwand für die Attacken um den Faktor 16 verringert werden. Da es sich bei dem Chip um einen Prototypen handelt, wurden auch Tests durchgeführt, welche zeigen dass der verwendete Chip den Spezifikationen entsprechend funktioniert.
Mit den durchgeführten Attacken konnte einmal mehr gezeigt werden, dass es nicht ausreichend ist einen sicheren Algorithmus zu verwenden. Auch bei der Implementierung muss darauf geachtet werden, dass Seitenkanalattacken verhindert werden. Die verwendeten Gegenmaßnahmen erschweren Seitenkanalattacken, jedoch müssen die Gegenmaßnahmen sorgfältig implementiert werden. Es soll für einen Angreifer nicht möglich sein, Informationen über die Parameter der Gegenmaßnahmen zu zu erhalten.

Schlüsselwörter: Nahfeldkommunikation, Seitenkanalattacken, Radiofrequenz-Identifikation

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Acknowledgements

First of all I would like to thank my supervisor Thomas Plos for the support and the valuable inputs during the work on this master thesis.
I also want to thank all the members of the Institute for Applied Information Processing and Communications (IAIK) for their hints and inputs during my work.
Most importantly I would like to thank my family for their support and encouragement.

Graz, September 2011 Thomas Korak

Contents

List of Tables

List of Figures

1. Introduction

Near-Field Communication (NFC) is a standardized wireless communication technology that works on short ranges. The rise of NFC-enabled mobile phones on the one hand and the usability of the technology on the other hand give this technology promising prospects for the future. There are many real-world examples where NFC can be used. Pay a bus fare or do other micropayments, enter or exit the office, log in to a computer or open the car doors are just some examples. In 2004 the so-called NFC Forum, a non-profit industry association, was formed which has now already 140 members. The members are from different fields (manufacturers, application developers, financial service institutions, ...) and work on the rise of the NFC technology. The main goals of the NFC Forum are to develop standard-based NFC specifications, encourage manufacturers to use NFC technology, ensure that NFC devices comply with the standards and train and educate enterprises in the field of NFC.
The main parts of an NFC-system are the reader and the transponder. The reader generates an electromagnetic field that is used for communication with a transponder. The transponders, which are also called tags, consist of a chip and an antenna. Tags are called passive if they extract the supply voltage from the reader field and active if they are battery powered.
On the first sight NFC might be seen as a secure communication technology because the communication range is only a few centimeters but there are several issues concerning security which have to be obeyed. First of all there is eavesdropping. If the communication is not encrypted an attacker can listen to the communication between two devices. For that purpose a small antenna has to be placed between sender and receiver. Data corruption, manipulation and insertion are also possibilities for an attacker to attack devices that use NFC technology. In order to address these security issues, cryptography has to be integrated into all components of NFC-systems, even in the tags. The Institute for Applied Information Processing and Communications (IAIK) together with austriamicrosystems [aus11b] and RF-iT-Solutions [iS11] have developed such a cryptographically protected tag chip (in the following called CRYPTA chip). The CRYPTA tag is a passive Radio-Frequency Identification (RFID) tag that works according to the NFC-Forum Type 4 tag specification. Key features of the chip are the NFC air interface that is compliant to the ISO14443A standard and the support of symmetric and asymmetric cryptography.
For enabling symmetric cryptography, the Advanced Encryption Standard (AES) is implemented on the chip. As AES is standardized by the National Institute of Standards and Technology (NIST) the interoperability with many other devices can be guaranteed. For asymmetric cryptography the Elliptic Curve Digital Signature Algorithm (ECDSA) is implemented. The ECDSA is a special form of the standardized Digital Signature Algorithm (DSA) using elliptic curves. Only a few prototype chips of the CRYPTA tag have been fabricated so far. For a prototype it is very important that all the implemented functions work according to the specification. The secure implementation of the cryptographic primitives is also of

great importance.
The first objective of this thesis is to test the correct execution of all the implemented functionalities. The focus in this part is put on the evaluation of AES and ECDSA, the evaluation of the protection mechanisms for the implemented file system as well as on the evaluation of the different clock frequencies the tag can operate with. Two RFID readers as well as an NFC-enabled mobile phone are used for the tests.
The second objective is to evaluate the security level of the AES implementation. So-called countermeasures are implemented on the tag to enhance the security level. We can evaluate the effectiveness of these countermeasures as they can be enabled and disabled. Side-channel analysis (SCA) attacks targeting this AES implementation are performed to reveal information about the secret key used for encryption and decryption. The power consumption during the encryption or decryption as well as the electromagnetic emanation of the chip are used as side channels. The power traces needed for the attacks are recorded using an oscilloscope and analyzed using statistical methods which are also presented in this work. After this analysis steps the correct values for the key bytes can be extracted with a high probability. The more traces are needed for a successful attack the higher is the complexity of the attack and the security level of the implemented cryptographic algorithm.
The remainder of the thesis is organized as follows:
Chapter 2 gives some background information to the CRYPTA chip. We focus on the applications where tags using this chip can be used as well as on the architecture of the chip.
Chapter 3 deals with the AES. First we present the algorithm for encryption as well as decryption followed by some attack scenarios.
Chapter 4 gives an introduction to the ECDSA and compares it to normal DSA. Here we show that the algorithm using elliptic curves is well-suited for applications with low hardware resources like RFID tags.
Chapter 5 gives an overview about SCA attacks. Here we present different approaches how side-channel information can be used in order to reveal some secret information of an implementation. In the end of the chapter we also give some countermeasures to make an implementation resistant against SCA attacks.
Chapter 6 points out the results of the functionality tests of the CRYPTA chip. We evaluated the commands for writing and reading files as well as the cryptographic commands. Also the results of the tests for the different configurations of the chip concerning the clock frequency or the data rates for communication with the different readers can be found in this chapter.
Chapter 7 deals with the results of the SCA attacks on the AES implementation of the CRYPTA tag. Here different methods for measuring the side-channel information are presented and an evaluation of the effectiveness of the implemented countermeasures is given. We also describe which steps are necessary in order to perform a successful attack.
Chapter 8 concludes the thesis and gives suggestions for future work.

2. The CRYPTA Tag

CRYPTA is the abbreviation for **CRY**ptographic **P**rotected **TA**g. It is a passive RFID tag with advanced cryptographic functions. CRYPTA is a government-funded project in cooperation with two companies: Austriamicrosystems and RF-iT Solutions. The CRYPTA tag was developed and evaluated at the Institute of Applied Information Processing and Communications (IAIK). The following chapter gives an overview of the tag with some application scenarios followed by a description of the architecture of the chip. Later in this chapter the focus is put on the development board where the chip is mounted on. Two devices were used for evaluation: the CRYPTA prototype tag as well as an FPGA prototype. The last section gives a description of the FPGA prototype that was used for evaluation.

2.1. Overview

The CRYPTA tag supports symmetric authentication using the Advanced Encryption Standard (AES) as well as asymmetric authentication using the Elliptic Curve Digital Signature Algorithm (ECDSA). For storing data on the tag an EEPROM with a size of 4096 bits is mounted on the chip. The EEPROM is organized in several files with different access protection mechanisms. The tag contains a capability container file (CC file) and a near-field communication (NFC) data-exchange format file (NDEF file) to be compliant to NFC forum Type-4 tags [NFC07]. It uses the ISO 14443A standard for communication with the reader. The protocol for anticollision and selection of the tag is implemented according to the ISO 14443-3 standard [Int01] and the advanced functionality is implemented according to the ISO 14443-4 standard [Int08]. Higher-layer information is transmitted using application protocol data units (APDUs). The ISO7816-4 standard [Int95] shows how the previously mentioned APDUs are constructed. The detailed specification of CRYPTA can be found in [Fel10].

2.2. RFID Systems

A typical RFID system consists of an RFID tag, a reader and a backend database. By using the reader field the tag and the reader can communicate with each other contactless. A tag is called passive if it extracts the supply voltage from the reader field. Active tags are equipped with a small battery for power supply, which allows to increase the communication range. The reader can be a dedicated hardware device connected to a computer as well as e.g. a smartphone like the Samsung Nexus S which has a build in NFC chip. When a large

number of mobile devices have NFC functionality integrated the applications are expected to become more and more complex. So also the need for security features rises. Because of that fact cryptographic primitives have to be implemented on RFID tags. There are two main constraints which have to be considered during the design of an RFID tag: the power consumption and the chip size. Especially for passive tags the power consumption has to be low because the amount of energy which can be extracted from the reader field is limited. The size of the chip is important because the costs of the chip increase with the size. An overview of the different RFID technologies is given in the 'RFID-Handbook' [Fin03].

2.3. Applications for CRYPTA

The CRYPTA tag is a passive tag that extracts the supply voltage from the reader field. No additional battery is needed to power the chip. Additionally, also the clock signal is extracted from the reader field. One of the major applications of CRYPTA is the proof of origin. Manufacturers can attach this chip to their (expensive) goods like designer clothes or watches and the customer can use the smartphone to check whether the article is original or a fake. The ECDSA is a way to perform such a proof of origin. It is an asymmetric-key system that uses a key pair. This key pair consists of a public key and a private key. The private key is stored on the tag and does not leave it. Additionally, the public key is stored on the tag which is used to verify the signature. The reader (smartphone) sends a challenge to the tag which signs this challenge using its private key. The generated signature is sent back to the reader where it can be verified using the public key. The authenticity of the public key can be verified by using a certificate which can be e.g. downloaded from the web site of the manufacturer. An introduction to public-key cryptosystems can be found in [RSA78] and a detailed description of the ECDSA is given in [JMV01].
Proof of origin can also be performed using symmetric cryptography. One disadvantage of this method is that the tag and the reader need to share the same secret key [HFP10]. Because of that fact it is easier for an attacker to reveal this secret key. Another disadvantage is that the manufacturer has to handle a lot of secret keys, one for each product group or even for each article. If an attacker can reveal the secret AES key fake products can be produced with a valid proof of origin. On the other hand if an attacker can reveal the private ECDSA key of one tag the manufacturer is able to revoke the certificate of the tag. In the case of revocation the verification of the public key certificate of the tag will fail. It is also possible to define a validity period for certificates. The certificates are only valid within this specific period.
On the CRYPTA tag the symmetric cryptography is mainly used to restrict read and/or write access to certain files. It is also needed during the personalization phase. Only authorized readers are allowed to perform the personalization. The following example shows the lifecycle of the CRYPTA tag.

During the production process the chip gets a unique ID and a temporary AES key. This AES key is for the personalization process and is shared between the chip manufacturer and the manufacturer who wants to protect the goods from imitating. These two files are the only valid files in the EEPROM when the tag leaves the chip manufacturer. All the other bits of the EEPROM are set to 0.

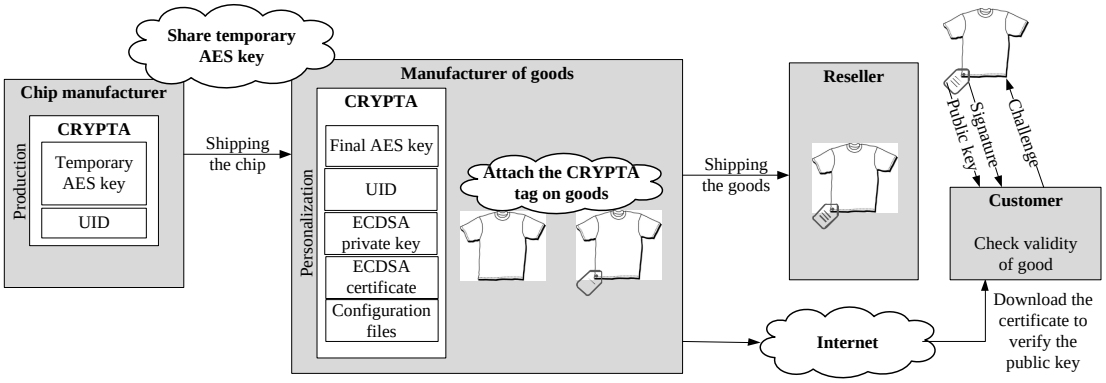

Figure 2.1.: Typical lifecycle of the CRYPTA tag.

The personalization is done by the manufacturer of the goods and works as follows: After a successful authentication using the temporary AES key, the ECDSA private key, the certificate containing the ECDSA public key, the configuration files as well as the final AES key are written to the EEPROM during personalization phase. Setting a special personalization bit finalizes this process. After that the tag can be attached to the goods which can be shipped to the resellers. The resellers sell the articles containing the proof of origin.
If a customer wants to buy a good equipped with a tag the validity can be verified. Figure 2.1 shows the path from the chip manufacturer to the customer.

2.4. Hardware Architecture

The main parts of the CRYPTA tag are the antenna and the chip. The chip can be split up into an analog part and a digital part. The analog part consists of the analog frontend. The digital part consists of three main components: the framing logic, the RFID control unit (RCU) and the crypto unit (c.f. Figure 2.2). The antenna is connected to the analog frontend which generates the supply voltage and the clock signal from the reader field. It is also responsible for the demodulation of data received from the reader and for modulation of data sent to the reader. The framing logic preprocesses received higher-layer data (ISO 14443-4) for the RCU according to the used standard. Received data from the lower layer (ISO 14443-3) is handled directly by the framing logic. The RCU is the main component of the tag. As one requirement during the planning phase was to build a flexible tag, the RCU contains a programmable microcontroller. The handling of the APDUs for higher layer data and command exchange from and to the reader is also implemented in the RCU. With the microcontroller the different files can be read from the EEPROM and the crypto unit can be controlled. The crypto unit is a special-purpose hardware supporting the Elliptic Curve Digital Signature Algorithm (ECDSA), the Advanced Encryption Standard (AES) and the Secure Hash Algorithm 1 (SHA-1). The focus when designing the crypto unit was on low power consumption and small chip area as these are the main constraints for a passive RFID

device.

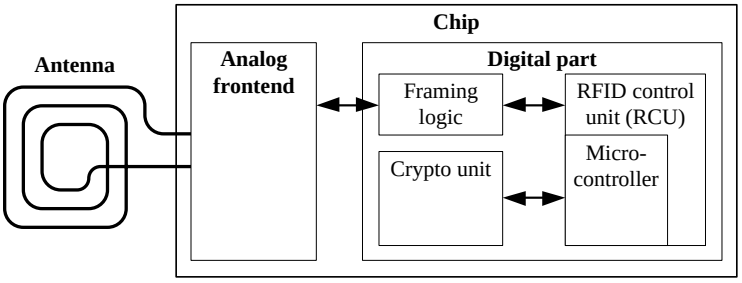

Figure 2.2.: The main components on the CRYPTA tag.

2.5. Crypto Unit

As mentioned in the previous section the requirements for a passive RFID device are low power consumption and small chip area. In order to meet these requirements a special crypto unit was designed for the CRYPTA chip. The target was to reuse several hardware modules (memory, control) for AES as well as ECDSA. The 8-bit microcontroller of the RCU controls the sequence of the algorithms of the crypto unit [HFW11]. In the next two sections we focus on the implementation of these two primitives.

2.5.1. AES Implementation

The AES implementation supports a fixed key size of 128 bits and encrypting as well as decrypting of data is possible. The 16x8-bit state of AES is expanded to a 16x16-bit state where half of it contains random data. The crypto unit processes the random values in the case of a dummy round. The insertion of dummy rounds is one of the two countermeasures to prevent side-channel analysis attacks (SCA) implemented on the CRYPTA chip. The second countermeasure is shuffling. In the case of shuffling the processing order of the bytes of the state is changed. An 8-bit value is used for the configuration of the countermeasures where the lower four bits specify the number of dummy rounds inserted at the beginning and the upper four bits define the order of the execution of the bytes in the state. More information on countermeasures can be found in the work of Mangard et al. [SM07] and in Section 5.5 of this work. Additional implementation details about the AES algorithm can be found in [HFW11] as well as in [FDW04].

2.5.2. ECDSA Implementation

The EDCSA implementation is based on the recommended NIST Weierstrass elliptic curve (EC) over $\mathbb{F}_{p192}$. A standardized curve is used to ensure the interoperability with other

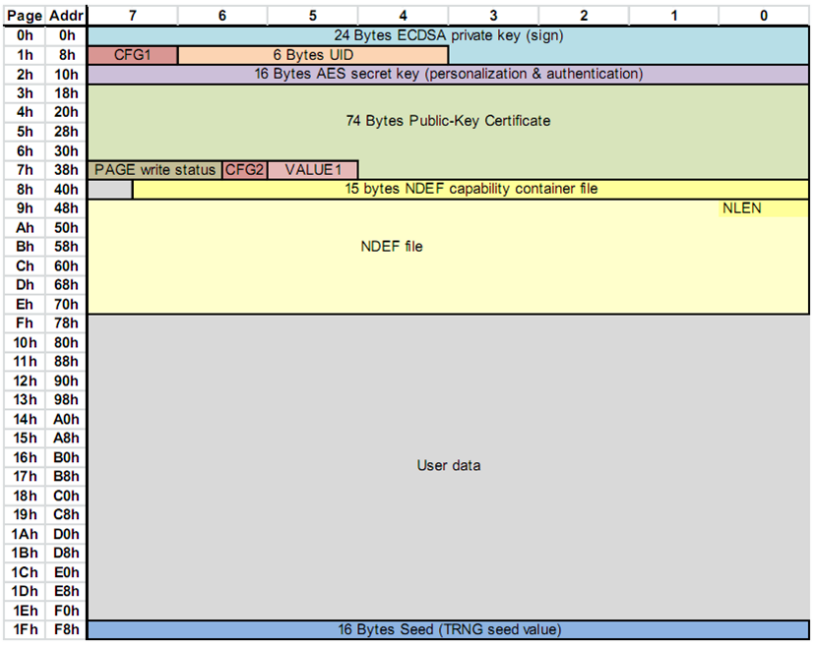

Figure 2.3.: EEPROM layout of the CRYPTA chip.

applications using the same curve and known optimizations for increasing the performance of the algorithm can be used (e.g. the NIST modular reduction [HMV04]). The used curve is analyzed well and it is considered to be secure. During the signature generation also the hash of the message has to be calculated using SHA-1. The hash algorithm implemented on the CRYPTA tag is designed for a fixed input length of 160 bits. With this limitation padding becomes very simple. In order to secure the signature generation the Montgomery ladder was used as scalar multiplication method. A detailed description of the implementation of the ECDSA can be found in the work of Hutter et al. [HFW11].

2.6. EEPROM

The following section gives a brief overview of the EEPROM of the CRYPTA chip. Non-volatile data (EC parameters, user data, keys) are stored on the EEPROM. It has a size of 4096 bits in total and is organized in words of 16 bits. Furthermore the EEPROM is partitioned in pages where every page has a size of eight words. At most one page (16 bytes) can be written during one write operation.

2.6.1. EEPROM Layout

The EEPROM is partitioned into 12 files with different access privileges. Figure 2.3 shows the layout in a graphical manner. Table 2.1 lists important features of the different files like file ID, size or read/write restrictions. The following restrictions exist to control reading

operations on files. No read access is allowed at all, read access is always allowed or read access is only allowed after a successful reader authentication (depending on the configuration of the chip). Several restrictions exist for writing the files. Some files can only be written during production and some can only be written during personalization. For other files a lock bit exists, where they can only be modified if this bit is not set. The CFG2 file and the Page RW file can only be modified once (one time programmable).

File Name	**File ID**	**Read**	**Write**	**Size**	**Description**
				Bytes	
User memory	0x0000	✓	∼	256	File for user data
Public key certficate	0x0001	✓	pers	74	Certificate string for ECDSA
NDEF file	0x0002	✓	∼	96	Contains information for NDEF application
ECDSA private key	0x0003	x	pers	24	The private key for ECDSA signature
AES secret key	0x0004	x	pers	16	The secret key for AES encryption
UID	0x0005	✓	prod	6	The unique ID of the tag
CC file	0xE103	✓	∼	15	The NDEF capability container file
CFG1	0x0006	✓	pers	2	Configuration file 1
VALUE1	0x0007	✓	✓*	2	Decrease-only value
CFG2	0x0008	✓	otp	1	Configuration file 2
Page RW status	0x0009	✓	otp	3	Lock bits for pages 0x09 to 0x1E
TRNG seed	0x000A	x	pers	16	Seed for true random number generator

✓ ... read access *always/after external authentication* allowed
x ... no read access
∼ ... write access if lock bit is not set
pers ... write access only during personalization
prod ... write access only during production
otp ... one time programmable
✓* ... write access only if new value is smaller

Table 2.1.: Features of the different files on the chip.

2.6.2. Configuration Files

In this section the files which are needed to configure the chip are explained.

Configuration Word 1 (CFG1)
With the CFG1 the clock frequency of the crypto unit can be configured. Also the write

and read access to the different locations in the EEPROM can be controlled. This allows to configure the tag such that a reader authentication is required for read and write operations to all files. A description of the bits of CFG1 can be found in Table 2.2.

Bit	Description
0 1	Configure the clock frequency of the chip 00...1.7 MHz, 01...848 kHz, 10...3.4 MHz, 11...6.8 MHz
2	Enable AES commands (1...disabled/0...enabled)
3	Enable ECDSA commands (1...disabled/0...enabled)
4	Enable NFC Type 4 mode (1...disabled/0...enabled)
5	Allow permanent disable after external authentication (1...do not allow/0...allow)
6	Write to EEPROM only after external authentication (1...enabled/0...disabled)
7	Read from EEPROM only after external authentication (1...enabled/0...disabled)
8	Write to Page RW status only after external authentication (1...enabled/0...disabled)
9-15	RFU

Table 2.2.: Description of the bits of CFG1.

Configuration Byte 2 (CFG2)
The CFG2 is one-time programmable. This means that bits that are set to 1 cannot be reset to 0. With the different bits of this byte the tag can be permanently disabled, it can be signalized if the item is sold and it can be switched from evaluation mode to standard mode. If the tag is in evaluation mode the countermeasures against SCA attacks are disabled. In the case of AES this means that no dummy rounds are inserted at the beginning and that the execution order of the bytes from the state is fixed. Setting bit 7 is the last step during personalization, it indicates that the tag is personalized. Setting the sold bit does not change anything in the tag's behaviour. It can be used for an anti-theft system to signalize if the customer has paid for the good. Table 2.3 gives a description of the bits of CFG2. Setting the bits of this configuration file has to be done very carefully because once set they cannot be reset. If for example the standard mode is once activated the mode cannot be changed to evaluation mode any more.

Bit	Description
0	Permanent disable (0...tag is enabled/1...tag is disabled)
1	Item is sold (0...item not sold/1...item is sold)
2-5	Not used, should be kept to 0
6	0...evaluation mode/1...standard mode
7	Personalization done (1...personalized/0...not personalized)

Table 2.3.: Description of the bits of CFG2.

Page RW Status

With the three bytes of the Page RW status file writing to the pages 0x09 to 0x1E can be allowed or not. If for example the first bit is set writing on page 0x09 is not allowed. As it can be seen on the layout of the EEPROM two files are affected from the lock bits, the NDEF file and the user data file.

2.7. Supported Commands

The focus in this section is on the description of the supported commands of the CRYPTA chip. Corresponding to the ISO 14443-4 standard the following block commands are supported: S-block commands, R-block commands and I-block commands. The correct coding of the different commands as well as examples can be found in [Int08]. The commands for selection and anticollision are according to ISO 14443A-3[Int01] and will not be explained in detail in this chapter.

2.7.1. S-Block Commands

There are two different S-block commands, one to increase the waiting time of the reader (waiting-time extension) and one to deselect the tag (deselect command):

- S(WTX)... waiting-time extension
  There is a specific time (frame waiting time FWT) that the reader waits for an answer of the tag before a timeout occurs [Int08]. If the tag performs a time-consuming calculation such as an ECDSA signature generation the FWT is not long enough. In that case the tag sends a S(WTX) to the reader to indicate that the next answer will not arrive within the FWT but within a temporarily-defined time. So a timeout can be prevented.
- S(DESELECT)... deselect command
  This command is sent by the reader to deselect a specific tag. If the CRYPTA tag receives the S(DESELECT) command it enters the halt state (HLTA).

2.7.2. R-Block Commands

The R-block commands are used to exchange acknowledgments between reader and tag. The acknowledgment always refers to the last received block. RACK indicates a positive acknowledgment and RNAK indicates a negative acknowledgment. In the case of an RNAK the last block is sent again.

2.7.3. I-Block Commands

The I-block commands are used to exchange higher-layer data between tag and reader. This higher-layer data is called application protocol data unit (APDU) and is coded according to the ISO7816-4 standard [Int95]. There is a different structure of the APDUs for requests from the reader to the tag and for responses from the tag to the reader.

Table 2.4 shows the structure of an APDU from the reader to the tag. The class byte CLA denotes the class of the following command (typically 0x00). The instruction to be performed by the tag is given by the instruction byte INS followed by the parameter bytes P1 and P2. With these to bytes parameters for the instruction can be set. L_c is called the length byte and denotes the length of the following DATA field. The DATA field is optional and if it is empty there is no L_c byte. The last field L_e is again one byte long and gives information about the expected length of the tag answer. If no data is expected from the tag no L_c byte is sent.

CLA	INS	P1	P2	L_c	DATA	L_e

Table 2.4.: APDU from the reader to the tag.

Table 2.5 gives an overview of the different instructions supported by the CRYPTA tag. With the SELECT command a file can be selected using the file ID or the file name. The selected file can be read using the READ BINARY command where P1 and P2 define the offset and L_e gives the number of bytes to read. With the UPDATE BINARY command the selected file can be modified. Again P1 and P2 are used to define the offset, L_c gives the number of bytes to write and the DATA field contains the data to write. There is one UPDATE BINARY command for personalization and one for standard operation. The two commands differ in their class byte. The INT AUTHENTICATE (internal authenticate) command is used to authenticate the tag to the reader using AES or ECDSA. There is a difference in the length of the challenge (AES: 8-byte long challenge, ECDSA: 16-byte long challenge) as well as in the expected length of the answer (AES: 16 bytes, ECDSA: 48 bytes). With the EXT AUTHENTICATE (external authenticate) command the reader authenticates to the tag using AES. Before sending an EXT AUTHENTICATE command, the reader has to request a challenge from the tag using the GET CHALLENGE command. This challenge is then encrypted using AES and sent to the tag in the EXT AUTHENTICATE command.

Command	**CLA**	**INS**	**P1**	**P2**	**L_c**	**DATA**	**L_e**
SELECT file	0x00	0xA4		0x00			
by ID			0x00		0x02	file ID	–
by name			0x04		0x07	file name	–
READ BINARY	0x00	0xB0	offset		–	–	#bytes
UPDATE BINARY		0xD6	offset		#bytes	data	–
standard write	0x00						
personalize	0x80						
INT AUTHENTICATE	0x00	0x88		0x00		challenge	
AES			0x00		0x08		0x10
ECDSA			0x01		0x10		0x30
GET CHALLENGE	0x00	0x84	0x00	0x00	–	–	0x80
EXT AUTHENTICATE	0x00	0x82	0x00	0x00	0x10	AES_K	–

Table 2.5.: The instructions supported by the CRYPTA tag.

Table 2.6 shows the structure of an APDU response from the tag to the reader. The DATA field is optional, depending on the instruction that the reader has selected in the previous

APDU. The length of the DATA field is L_e. The status of the performed instruction is indicated with a status word which consists of the two bytes SW1 and SW2. With the status word `0x9000` (SW1 = `0x90`, SW2 = `0x00`) the tag indicates that the requested instruction has been performed successfully. The meaning of other status words can be found in Table 2.7.

DATA	SW1	SW2

Table 2.6.: APDU from the tag to the reader.

Status word (SW1‖SW2)	**SELECT**	**READ BINARY**	**UPDATE BINARY**	**INT AUTHENTICATE**	**GET CHALLENGE**	**EXT AUTHENTICATE**	
0x9000	x	x	x	x	x	x	OK
0x6A82	x						File/application not found
0x6982	x	x	x	x			Security status not satisfied
0x6D00	x	x	x	x	x	x	Command not supported/invalid
0x6282		x					End of file reached
0x6700			x				Wrong length
0x6B00			x				Wrong Parameters
0x6400			x	x	x		Power-check fail
0x6581			x				Memory failure during writing
0x6A80				x			Wrong parameters

Table 2.7.: Status words of the different commands and their meaning.

2.8. CRYPTA-Chip Test Board

The CRYPTA chip is a prototype and it is mounted on a test board. One advantage of the test board is that the EEPROM content can be changed without any restrictions (e.g. the bits of CFG2 can be set from 1 to 0 again) because there is direct access to the EEPROM via a serial debug interface. The interface is accessible as 10-pin header on the test board and can be connected to a programming board. This programming board is connected to a computer via a USB cable. A special software is used to read and write the content of the EEPROM. Each word can be written and read separately without any restrictions.
Another advantage of the programming board is that it allows to power the tag with an

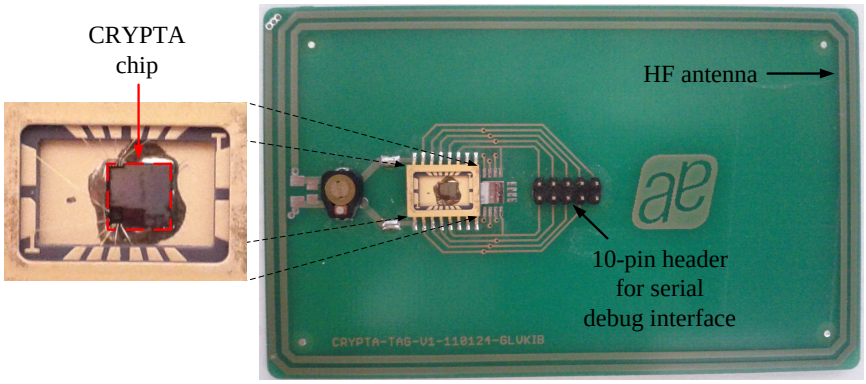

Figure 2.4.: The CRYPTA tag chip with test board for evaluation.

external power supply. In that case the tag does not have to generate the supply voltage from the reader field. This gave us the ability to measure the power consumption of the chip with a resistor in the ground line. Figure 2.4 shows the test board with the chip mounted on it. The size of the board is about 90 mm x 55 mm and is close to the standardized ID-1 format (commonly used for banking cards) of 85.725 mm x 53.975 mm [Int11].

2.9. FPGA Prototype

This section gives a short overview of the FPGA prototype of the CRYPTA tag which was used for SCA attacks in order to evaluate the countermeasures. The FPGA prototype is implemented on the IAIK HF FPGA DemoTag. The DemoTag is a programmable RFID tag which appears to the reader as a passive tag. It is convenient for early prototypes of RFID tags in order to test the functionality. It consists of two parts: The motherboard and the FPGA board. The first part is the so-called motherboard (cf. Figure 2.6), where the antenna coil as well as the analog frontend is placed. Also connectors for the power supply are placed on the motherboard. The second part is the board with the FPGA (cf. Figure 2.5) which can be plugged on top of the motherboard. The Xilinx Spartan-3 XC3S1000 [Xil08] FPGA is used on the DemoTag. Although the FPGA DemoTag only works with an external power supply (does not extract the power supply from the reader field) it behaves like a passive RFID tag. The analog frontend extracts the clock signal from the reader field and provides it to the FPGA.
The FPGA implementation of the CRYPTA chip behaves like the real chip which is mounted on the test board. Two errors were found on the real chip (see Chapter 6 and Chapter 7) which were corrected in the FPGA implementation. The impact of the corrections could be tested this way without fabricating a new chip.
More technical details about the HF FPGA DemoTag can be found in the work of Feldhofer et. al [FAH+10].

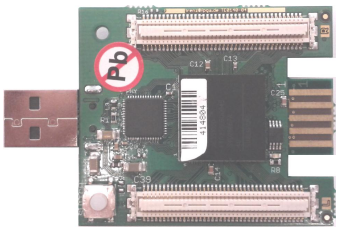

Figure 2.5.: FPGA board.

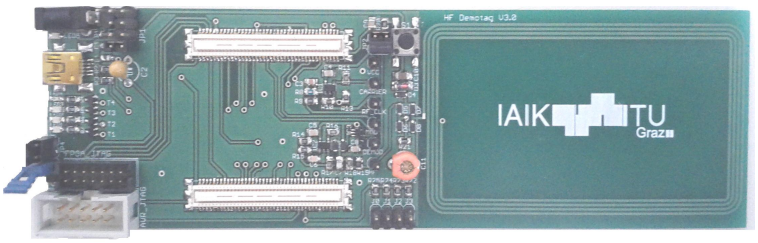

Figure 2.6.: The motherboard.

2.10. Summary

In this chapter we gave a short overview of the project CRYPTA. The main goal of the project is to enable advanced cryptography on passive RFID tags. We first gave an overview of the project followed by some real-world applications. Then the architecture of the chip was described with the focus put on the cryptographic unit and the EEPROM module. Next the supported commands of the chip were explained to enable a reader to communicate with the CRYPTA tag. In the end we discussed the hardware for the SCA attacks presented in Chapter 7, on the one hand the CRYPTA chip test board and on the other hand the FPGA prototype.

3. An Introduction to AES

The Advanced Encryption Standard (AES) is the standardized version of the Rijndael algorithm, which is a symmetric block cipher for data encryption and data decryption. The algorithm is standardized in the FIPS-197 document of the National Institute of standards and Technology (NIST) which can be found at [oSN01]. The differences between AES and Rijndael are the different values of supported block length and different possible key lengths described in [JD02].
In the following chapter a short introduction to AES is given. First the parameters for the algorithm are explained. Then the focus is put on the encryption and decryption processes as well as on the transformations used by the algorithm. At the end of this chapter we explain two approaches for attacking implementations of AES.

3.1. Parameters for AES

As input for encryption, a plaintext block with a length of 128 bits and a key are used. Three different key lengths are supported: 128 bits (AES-128), 192 bits (AES-192) and 256 bits (AES-256). The output of an encryption is the ciphertext block with a length of 128 bits.
For decryption, a ciphertext block serves as the input for AES together with the key (again the same three key lengths as for encryption are supported) that has been used for encryption of the plaintext. The output here is again the original plaintext block [oSN01].
If the plaintext block is shorter than 128 bits a padding scheme has to be used to expand the plaintext to 128 bits. If the message length is greater than 128 bits then a block-cipher mode of operation has to be used. Several so-called modes of operation can be found in [Dwo01].

3.2. Encryption Process

AES consists of a defined number N_r of round transformations on the so called state that depends on the key length N_k. The size of the state is 128 bits (N_s=16 bytes) which is organized as a 4x4 matrix. For N_k=16 bytes there are N_r=10 rounds, for 24 bytes 12 rounds and for 32 bytes 14 rounds. Additionally there is one initial round at the beginning which does not add to the number of rounds. The final round is also a bit different than the other rounds. One round transformation is skipped in this final round as it can be seen later in this chapter.
All the other rounds consist of the following four transformations: `subBytes()`, `shiftRows()`, `mixColumns()` and `addRoundKey()`. For every round a new round key is extracted from the cipher key.

- The transformation `subBytes()` is a byte substitution. It is a nonlinear operation and can be implemented as a table lookup. This transformation is applied bytewise as it can be seen in Figure 3.1. The whole S-box table can be found on page 16 of [oSN01].
- The transformation `shiftRows()` shifts the rows of the state matrix. Different rows are shifted with a different offset as shown in Figure 3.2.
- The function `mixColumns()` transforms the state colum by colum, as it can be seen in Figure 3.3. The transformation can be written as a matrix multiplication.
- The `addRoundKey()` transformation is the last transformation of every round. Every byte of the state is XORed with the corresponding byte of the round key.

Listing 3.1 shows the pseudocode of an AES encryption with a key length of 128 bits (10 rounds). As it can be seen the initial round only consists of a key addition with the original key and in the final round the `mixColums()` transformation is left out. For every round a new round key is generated. A detailed description of the key expansion function can be found in [JD02] as well as in [oSN01].

```
1  ciphertext AESEncrypt(plaintext, aeskey)
2  {
3    load plaintext into state;
4    state = AddRoundKey(aeskey);
5
6    for rnd = 1 to 9
7    {
8      key = expandRoundKey(aeskey, rnd);
9      state = subBytes();
10     state = shiftRows();
11     state = mixColumns();
12     state = addRoundKey(key);
13   }
14   key = expandRoundKey(aeskey, 10);
15   state = subBytes();
16   state = shiftRows();
17   state = addRoundKey(key);
18
19   ciphertext = state;
20 }
```

Listing 3.1: Pseudocode for an AES-128 encryption.

3.3. Decryption Process

For decrypting data the inverse cipher with the modified transformations `invSubBytes()`, `invShiftRows()` and `invMixColumns()` as well as `addRoundKey()` with the inverse key sched-

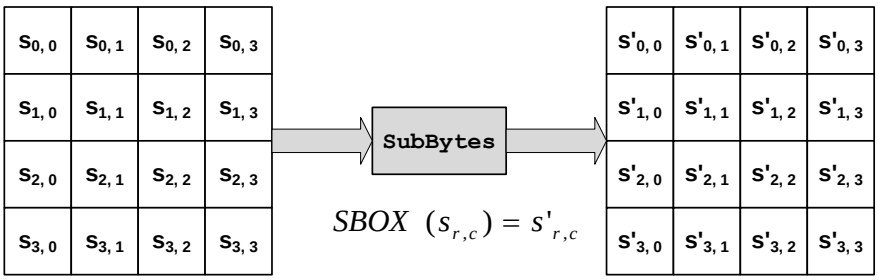

Figure 3.1.: The `subBytes()` transformation [oSN01].

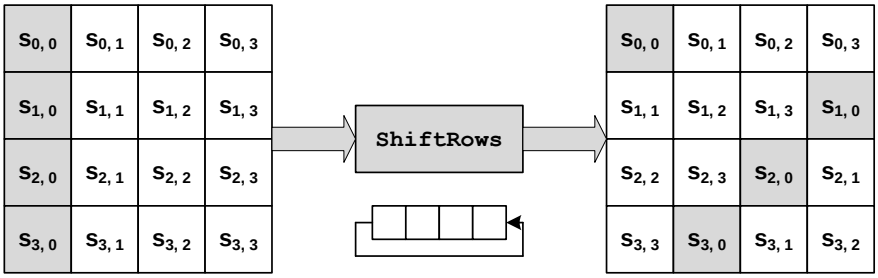

Figure 3.2.: The `shiftRows()` transformation [oSN01].

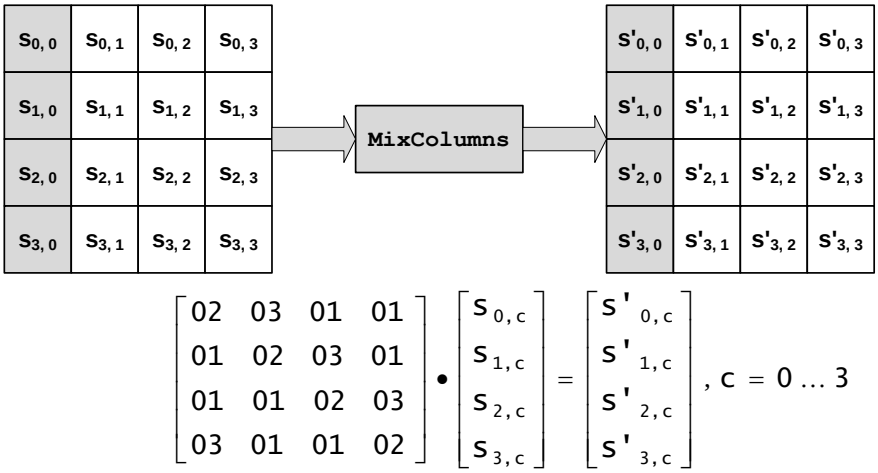

Figure 3.3.: The `mixColumns()` transformation [oSN01].

ule has to be used to get the plaintext again.
In [JD02] two different solutions of the decryption algorithm are shown, the straightforward decryption algorithm and the equivalent decryption algorithm. The difference between the two approaches is the order of the transformations. The straightforward approach simply inverses the encryption algorithm, as it can be seen in the pseudocode in Listing 3.2. Using the equivalent approach it is possible to put the `invSubBytes()` transformation at the beginning of each round again as it can be seen in Listing 3.3. For hardware implementations it is more convenient to have the nonlinear transformation at the beginning of each round. The type of implementation of the AES decryption (straightforward or equivalent) does not make any difference for side-channel analysis attacks (SCA). For the equivalent decryption algorithm also the key expansion function has to be adapted (`eqExpandRoundKey()`).
A detailed description why and how the order of the transformations can be changed is given in Chapter 3 of [JD02].

- The transformation `invSubBytes()` is again a table lookup in the inverse substitution box. This inverse substitution box can be found on page 22 in [oSN01].
- The transformation `invShiftRows()` cyclically shifts the second, third and fourth row of the state for one, two and three positions. The shift direction is now opposite to the direction of `ShiftRows()`.
- The transformation `invMixColumns()` works on each column separately just like `mixColumns()` but using a different matrix. The transformation as well as the matrix can be seen in Fig. 3.4.

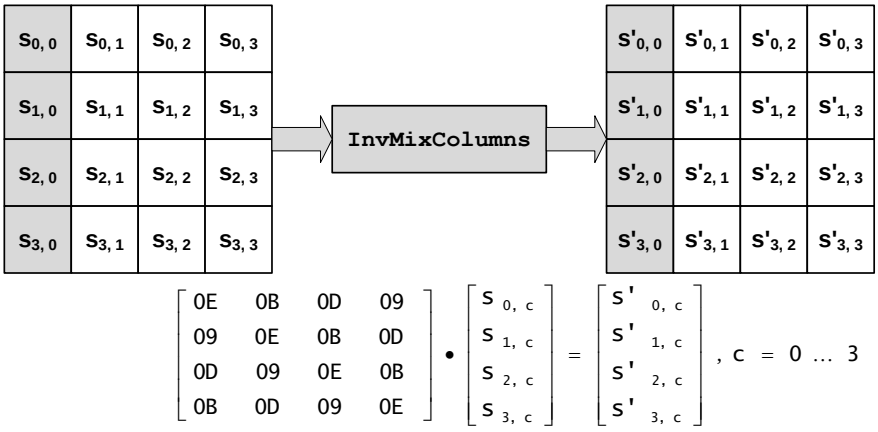

Figure 3.4.: The `InvMixColumns()` transformation [oSN01].

```
1 plaintext AESDecryptStraightforward(ciphertext, aeskey)
2 {
3   load ciphertext into state;
4
5   key = expandRoundKey(aeskey, 10);
6   state = addRoundKey(aeskey);
```

```
  state = invShiftRows();
  state = invSubBytes();

  for rnd = 9 downto 1
  {
        key = expandRoundKey(aeskey, rnd);
        state = AddRoundKey(key);
        state = invMixColumns();
        state = invShiftRows();
        state = invSubBytes();
  }
  state = addRoundKey(aeskey);
  plaintext = state;
}
```

Listing 3.2: Pseudocode for AES-128 straightforward decryption.

```
plaintext AESDecryptEquivalent(ciphertext, aeskey)
{
  load ciphertext into state;

  key = eqExpandRoundKey(aeskey, 10);
  state = addRoundKey(key);

  for rnd = 9 downto 1
  {
        key = eqExpandRoundKey(aeskey, rnd);
        state = invSubBytes();
        state = invShiftRows();
        state = invMixColumns();
        state = AddRoundKey(key);
  }
  key = eqExpandRoundKey(aeskey, 0);
  state = invSubBytes();
  state = invShiftRows();
  state = AddRoundKey(key);
  plaintext = state;
}
```

Listing 3.3: Pseudocode for AES-128 equivalent decryption.

3.4. Attacks on AES

This section presents some approaches to attack AES implementations using side-channel information. As side-channel information e.g. the power consumption or the electromagnetic

radiation of the cryptographic device can be used.
For a differential power analysis an intermediate result of the AES algorithm has to be found which suits for the attack. This intermediate result has to be a function *f(d, k)*, where d is a known value and k is a small part of the key.
In the first approach, the value d is a part of the plaintext. The idea here is to attack the first round of AES which can be seen in Figure 3.5.
As mentioned above d is one byte of the plaintext and k are the different key guesses. Every key guess has the size of one byte. So 256 key guesses are needed to cover the key space for one key byte. In order to reveal the whole AES secret key the attack has to be done for all the 16 plaintext bytes as a successful attack using the first plaintext byte reveals the first byte of the secret key and so on. The structure of f can be seen in Equation 3.1 as well as in Figure 3.6.

$$f(d, k) = \text{S-box}(d \oplus k) = after\_sbox \tag{3.1}$$

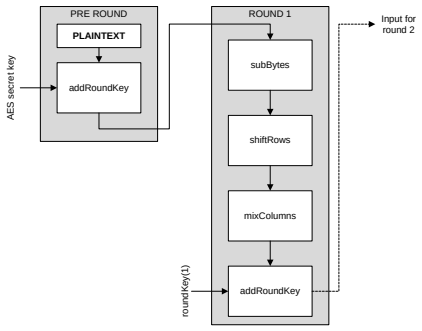

Figure 3.5.: Pre-round and first round of AES.

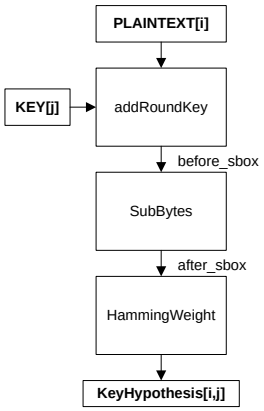

Figure 3.6.: Attack on the first round of AES.

The next approach is to attack the last round of AES. In this case the value d is a part of the ciphertext. The result of this attack is not the secret AES key but the last round key. Using the inverted key schedule algorithm the secret AES key can be computed with this result. Figure 3.7 shows the last two rounds of AES.
As the ciphertext is known and the 10^{th} round key serves as the value k for the formula f in Equation 3.2, the input of the 10^{th} round can be used as the intermediate result. The calculation of this intermediate result can be seen in Figure 3.8. The inverse of the `addRoundKey()` transformation is again `addRoundKey()` as it is just an XOR operation. As the inverse of the `SubBytes()` transformation `invSubBytes()` can be used, which is also used for decryption. The `shiftRows()` transformation can be left out because it is not relevant for the attack but it has to be obeyed when allocating the results to the key bytes. The `mixColumns()` transformation does not play a part in this attack because it is not used in the final round.

$$f(d,k) = \text{S-box}(d \oplus k)^{-1} = before\_sbox \tag{3.2}$$

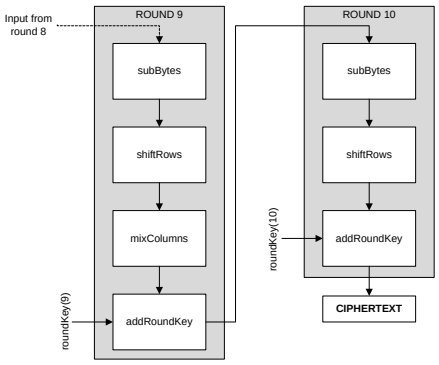

Figure 3.7.: Last two rounds of AES.

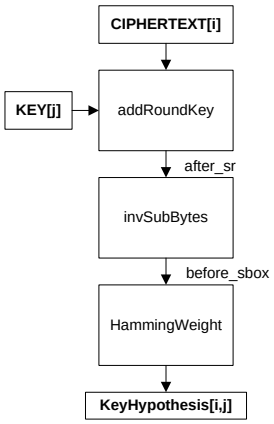

Figure 3.8.: Attack on the last round of AES.

4. An Introduction to ECDSA

This chapter gives a brief introduction to digital signatures with the focus put on the Elliptic Curve Digital Signature Algorithm (ECDSA) which is supported by the CRYPTA tag. Digital signatures enable a person to sign digital forms and to send the signed forms via e.g. e-Mail. It is not necessary to print out the form to attach the handwritten signature. In order to ensure a secure signature generation and signature verification public-key cryptography is used [Aus11a]. Public-key cryptography (asymmetric cryptography) uses a key pair consisting of a private key and a public key. The two keys are mathematically related to each other and if only the public key is known the private key cannot be derived. A one-way trapdoor function f is the basis for this system. With such a function it is easy to calculate $y = f(x)$ if x is known. If only y is known it is hard to find the appropriate x. The trapdoor is the key which makes it easy to calculate $x = f^{-1}(y)$. With asymmetric cryptography confidentiality, integrity, authenticity as well as nonrepudiation can be ensured. If Bob wants to send a secret message to Alice over an insecure channel he can encrypt the message using Alice's public key. Only Alice can decrypt the message with her private key again, so the confidentiality of the message is ensured. In order to ensure integrity, authenticity and nonrepudiation Bob has to encrypt (sign) the hash value of the message he wants to send to Alice using his private key. The hash value is a fixed-length value and it should be practically impossible to find a second (meaningful) message having the same hash value. Alice can also calculate the hash value of the received message. As Alice knows the public key from Bob she can decrypt the signed hash value in order to compare it with the hash value she has calculated. If the values are equal Alice can be sure that nobody has changed the message (integrity) that the message comes from Bob (authenticity) and Bob cannot deny that he has signed the message (nonrepudiation). The encrypted hash value is also called the digital signature of the message. Figure 4.1 shows the different steps required for signature generation and verification. They are the same for the presented schemes, just the mathematics behind the schemes are different.
Different schemes can be used to generate digital signatures. In the next section there is a short introduction to the Digital Signature Algorithm (DSA). It is the predecessor of the ECDSA which is described one section later. Both algorithms are standardized by the National Institute of Standards and Technology (NIST) in the Federal Information Processing Standard Publication (FIPS PUB) 186-3 [Nat09]. The need for the introduction to the DSA rises because in the end of this chapter we make a comparison of the two algorithms. We show that the ECDSA is the preferred choice for devices where hardware resources like memory are limited.

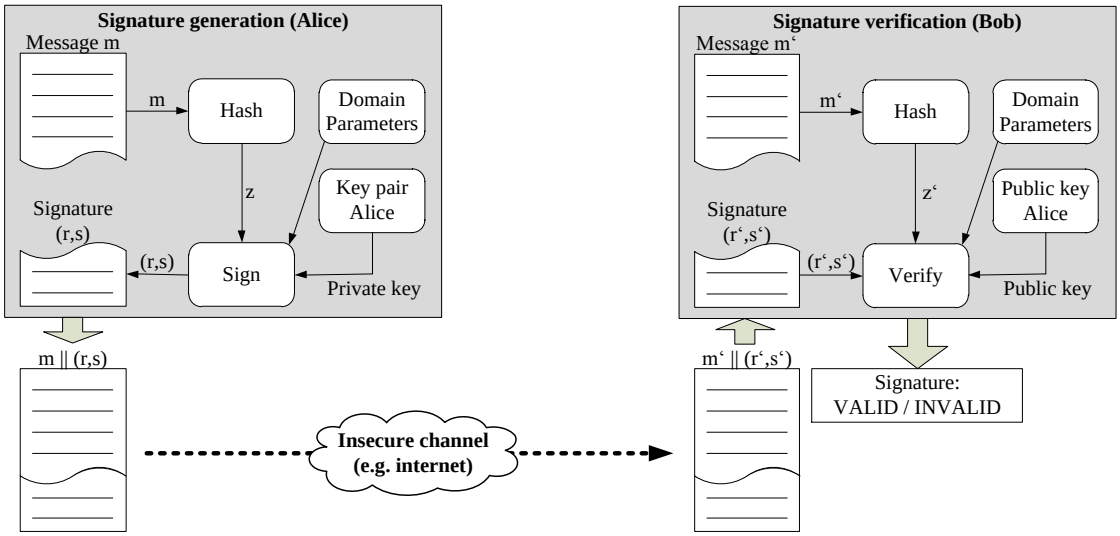

Figure 4.1.: The steps during a signature generation by Alice and a signature verification by Bob.

4.1. The Digital Signature Algorithm

The DSA was proposed in 1991 by the NIST as the first digital-signature scheme recognized by a government. It is a variant of the ElGamal signature scheme and the underlying problem is the discrete logarithm problem DLP [Sti02]. The discrete exponentiation is the one-way trapdoor function in this case. If there is a cyclic group G with a generator x, it is easy to calculate $y = f(x) = x^d mod(N)$. The inverse function $x = f^{-1}(y)$ to get x if only y is known is a hard mathematical problem and is called the discrete logarithm. If the inverse function is equipped with some extra knowledge (the value e) it is easy to calculate x again: $x = f^{-1}(y, e) = y^e mod(N)$.
In order to generate a digital signature using the DSA several domain parameters have to be created at the beginning. The parameters are: A prime modulus p, a prime divisor q, a generator g of the subgroup $p\, mod(q)$, a randomly chosen private key x, a public key y which is mathematically related to x and a secret number k. The value of k is randomly chosen and it has to be unique for each signed message. A description of the values can be found in Table 4.1.

Algorithm 1 shows how signing the message m works. The value pair (r, s) denotes the signature of the hash value z of m (The Secure Hash Function (SHA) [Nat08] is used to get the hash value of the message):

There is a three-step process to verify the received signature. The received values for m, r and s are m', r' and s'. In the first step it is checked if r' and s' are in the correct borders. In the second step several intermediate values are calculated, also the hash value of the message m' is calculated in this step. In the last step the received value r' is compared to a

p	prime modulus L... bit length of p	$2^{L-1} < p < 2^{L}$
q	prime divisor of (p-1) N... bit length of q	$2^{N-1} < q < 2^{N}$
g	generator of the subgroup of order q mod p	$1 < g < p$
x	private key, randomly chosen	$0 < x < q$
y	public key	$y = g^{x} mod\, p$
k	secret number, unique to each message	$0 < k < q$

Table 4.1.: Domain parameters of the DSA.

Algorithm 1 DSA signature generation

$r = (g^{k} mod\, p) mod\, q$
$z = HASH(m)$
$s = (k^{-1}(z + xr)) mod\, q$

calculated value v and if they are equal the signature is valid. Algorithm 2 shows in detail how a signature verification works.

Algorithm 2 DSA signature verification

if $(0 < r' < q)\,\&\,(0 < s' < q)$ **then**
  $w = ((s')^{-1}) mod\, q$
  $z' = HASH(m')$
  $u1 = (z'w) mod\, q$
  $u2 = ((r')w) mod\, q$
  $v = ((g^{u1} y^{u2}) mod\, p) mod\, q$
  if $v == r'$ **then**
    The signature is valid
  else
    The signature is invalid
  end if
else
  The signature is invalid
end if

If the signature validation is successful the receiver can be sure that the message comes from the expected sender and has not been modified during transmission. A detailed description of the algorithm can be found in [Nat09].

4.2. The Elliptic Curve Digital Signature Algorithm

The ECDSA is a signature scheme using elliptic curves. Elliptic curve cryptography (ECC) was invented in 1987 by Koblitz [Kob87]. An elliptic curve is a set of points. These points

q	field order	
FR	field representation for $\mathbb{F}_q$	
a, b	coefficients defining the curve	$a, b \in \mathbb{F}_q$
P	base point	$P = (x_P, y_p) \in E(\mathbb{F}_q)$
n	The order of P	
h	The cofactor	$h = \#E(\mathbb{F}_q)/n$
d	The private key	$0 < d < n$
Q	The public key	$Q = dP$

Table 4.2.: Domain parameters ECDSA.

are defined by the equation $y^2 = x^3 + ax + b$ where $a, b \in \mathbb{F}_p$ and $4a^3 + 27b^2 \neq 0 \, mod(p)$. A pair (x, y) is a point on the curve if (x, y) satisfies the upper equation and $x, y \in \mathbb{F}_p$. The addition of two points as well as the multiplication of a point with a scalar value are defined and the result of a calculation is again always a point on the curve. In ECC there exists a discrete logarithm problem as well, which is called the elliptic curve discrete logarithm problem (ECDLP) [HMV04]. It is assumed to be harder compared to the DLP over a finite field. The ECDLP is defined as follows: Having an elliptic curve E over $\mathbb{F}_q$ and a point P of order n on the curve E. Also the point Q is known which satisfies: $Q = lP$. Find the integer l. The book [HMV04] gives a detailed mathematical introduction into the field of elliptic curve cryptography.
As in case of the DSA several domain parameters have to be created when using the ECDSA. The parameters are the following: The field order q, a field representation FR of the underlying field $\mathbb{F}_q$, the coefficients a and b which define the curve, a base point P with order n defining the security level, a cofactor h, the private key d and the public key Q. A description of the values can be found in Table 4.2.
Algorithm 3 shows how the signature generation for a message m works. The signature is the value pair (r, s). In order to get the hash value of the message an approved hash function from [Nat08] shall be used.
In order to verify the signature of a received message several steps have to be performed. The received values for m, r and s are m', r' and s'. The verification process is quite similar to the DSA. In the first step it is checked if r' and s' are within the correct borders. In the second step several intermediate values have to be calculated. If the calculated point $X \neq \infty$ the x-coordinate x_1 of this point is calculated. The signature is valid if $x_1 mod(n)$ is equal to the received value r'. In Algorithm 4 a detailed description of the verification process can be found.

Algorithm 3 ECDSA signature generation

$k \in [1, n-1] \, random$
$kP = (x_1, y_1)$
$r = x_1 mod \, n$
$z = HASH(m)$
$s = k^{-1}(z + dr) mod \, n$

If the signature verification is successful the receiver can be sure that the message comes from

Algorithm 4 ECDSA signature verification

if $(0 < r' < n) \,\&\, (0 < s' < n)$ **then**
  $z' = HASH(m')$
  $w = s^{-1} mod\, n$
  $u_1 = (z'w) mod\, n$
  $u_2 = ((r')w) mod\, n$
  $X = u_1 P + u_2 Q$
  if $X \neq \infty$ **then**
    Get the x-coordinate x_1 of the point X.
    $v = x_1 mod(n)$
    if $v == r'$ **then**
      The signature is valid
    else
      The signature is invalid
    end if
  else
    The signature is invalid
  end if
else
  The signature is invalid
end if

the expected sender and that the content has not been modified. A detailed mathematical description of the ECDSA as well as several security considerations can be found in the book of Hankerson et. al [HMV04].

4.3. Comparison between the DSA and the ECDSA

In this section we want to compare the DSA and the ECDSA. Whenever a public-key scheme is selected for a specific application several points have to be considered. The most important points are: functionality, security, performance and resource consumption. Functionality simply means that the scheme has to fulfill all the requirements of the application. The hardness of the underlying problem (e.g. DLP, ECDLP) can be used to evaluate the security of the scheme and the runtime can be taken as a performance measure. The amount of required memory can be used to evaluate the resource consumption of a scheme.

	Security level (bits)				
	80	112	128	192	256
ECDSA parameter n	160	224	256	384	512
DSA parameter p	1024	2048	3072	8192	15360

Table 4.3.: Comparison of key sizes (in bits) for different security levels [HMV04].

As it can be seen in Table 4.3 the parameter n for ECDSA is a lot smaller compared to the parameter p for DSA for the same security level. The parameter n for ECDSA also grows much slower for an increasing security level. The compared parameters are the limits for the key sizes and if these parameters are small the memory requirements decrease. So the ECDSA is well suited for applications with low (memory) resources like RFID tags. Also the size of the numbers that are handled during the calculation of the signature is smaller for the signature scheme based on elliptic curves.
Calculations on elliptic curves are more complex compared to calculations in finite fields but in the recent years several algorithms have been proposed to increase the efficiency of the calculation. Several approaches have also been published where the ECDSA signature generation is realized in special-purpose hardware as it is done on the CRYPTA chip. Some of these approaches can be found in the works of Aigner et al. [ABHW04], Furbass et al. [FW07], Wenger et al. [WFF10] and Hutter et al. [HFW11].

5. Attacking Cryptographic Devices

This chapter gives an introduction to attacks on cryptographic devices. Such devices can for example be security-enabled smartcards as well as a software implementation of a cryptographic primitive on a computer or smartphone. A special class of such attacks on devices are so called side-channel analysis (SCA) attacks. These attacks use properties like power consumption or timing behaviour of cryptographic devices to get e.g. information about the used secret key. Side-channel analysis attacks were first introduced by Kocher in [Koc96]. In this article the author describes how to reveal details about the secret key in RSA or the fixed exponents in Diffie-Hellman cryptosystems by measuring the runtime of key-dependent parts of the algorithm.
The following chapter first gives an overview of different types of attacks on cryptographic devices. After that the focus is put on side-channel analysis attacks and here in special on power analysis. The next sections will give information about the recording of power traces as well as providing details about statistical properties of power traces. After that there is a description of simple power analysis followed by the description of differential power analysis. In the end of this chapter countermeasures against power analysis attacks are given.

5.1. Types of Implementation Attacks

Implementation attacks can roughly be divided into the two groups as stated in [SM07]: passive attacks and active attacks.

- **Passive Attacks** For this attack type the cryptographic device is operated inside its specifications. That means e.g. the voltage limits of the power supply or the maximum clock frequency are not exceeded. So the attacked device is operated in its normal environment under normal conditions.
- **Active Attacks** In this type of attacks the attacker is given plenty of scope in attacking the cryptographic device. Inputs, outputs and even the environment can be manipulated for the purpose of the attack by the adversary.

Another distinction can be made by the fact if the cryptographic device is depackaged during the attack or not and up to which extent the depackaging process is done. In that case [SM07] distinguishes three different types of attacks: non-invasive attacks, semi-invasive attacks and invasive attacks.

- **Non-Invasive Attacks** Non-invasive attacks do not depackage the cryptographic device. The device is attacked as it is and so these attacks seem to be a real threat for practical applications. An attacker can attack the device and after revealing e.g. the secret key give it back to the victim, who does not notice that the device has been attacked. The group of non-invasive attacks are also known as side-channel analysis attacks as different side channels for this attacks can be used. The most popular ones can be found in the numeration below.
- **Semi-Invasive Attacks** For conducting semi-invasive attacks single chips of the cryptographic device may also be opened but in contrast to the upper type there is no contact between measuring device and chip. With a special probe (EMR probe) the electromagnetic emanation of the chip can be measured without direct contact to the surface.
- **Invasive Attacks** At invasive attacks the attacker has the full control over the cryptographic device which is attacked. During this attack the cryptographic device is depackaged to measure signals between different components. It is also possible to open a single chip of the device to get direct access to the chip surface. So even chip-internal signals can be measured or altered. The attack is called passive if only signals are measured and active if internal signals are altered to cause a faulty behavior.

The following list gives a summary of popular side channels which can be used to attack a cryptographic device. More detailed information of attack scenarios can be found in [Iva05].

- **Timing Attack** These type of attacks use the timing behavior of a device to reveal e.g. the used secret or private key. One example of such an attack is described in [Koc96]. Here the author takes advantage of the fact that the duration of the modular exponentiation $R = y^x mod(n)$ is dependent on the used exponent x (x is the secret key).
- **Cache Attack** Cache attacks take advantage of the fact that on modern processors more processes are able to run more or less concurrently. As the memory of the processor (the so called cache) is a limited resource, different processes have to share this memory. The data stored for every process in the cache is protected by virtual protection mechanisms but the metadata (e.g. access patterns) are not well or not at all protected. So an attacker is able to observe the cache usage of the desired process to get some secret information about it. A detailed description of cache attacks can be found in [OST05].
- **Fault Analysis** In the case of fault analysis the attacker tries to get information from the implemented algorithm by intentionally introducing faults and analyzing the output. There are different ways to generate erroneous behavior in the cryptographic device. The most popular ways are to try out not-specified input parameters, use spike attacks (spikes in the power supply lines), glitch attacks (change clock frequency or amplitude for a short time) or optical attacks (alter bits by applying light).
- **Differential Fault Analysis** Differential fault analysis was introduced by A. Shamir et al. in [BS96]. The attack described there was the first one targeting secret key

cryptosystems on smartcards. The idea of the attack is that the attacker knows if a fault occured during encryption but not where. The attacker is able to encrypt the same plaintext twice getting one correct and one incorrect ciphertext. When doing this step several times and comparing the two different ciphertexts belonging to the same plaintext it is possible to reveal the used secret key.

- **Simple Power Analysis** Simple power analysis (SPA) means that the attacker has only one (single-shot SPA attacks) or a small number (multiple-shot SPA attacks) of recorded power traces of a cryptographic device during encryption. So a lot of knowledge about the implemented algorithm is needed in order to reveal the secret key with a low number of power traces. A detailed description as well as examples and approaches for SPA can be found in Section 5.3.
- **Differential Power Analysis** Differential power-analysis (DPA) attacks need in contrast to SPA attacks a large set of power traces to find parts of the secret key of the cryptographic device. The advantage of DPA attacks on the other hand is that there is no detailed knowledge about the implemented algorithm needed. SPA attacks analyze one or a small set of power traces along the time axis, in DPA attacks the power consumption of many traces at fixed points in time is analyzed. Examples as well as a detailed description on DPA attacks can be found in section Section 5.4.

5.2. Power Traces

As mentioned in the previous section the focus of this work is on power analysis attacks. To perform such attacks one needs to record power traces of the cryptographic device during the encryption process. The number of traces needed for the attack depends on the type of the attack (SPA or DPA) as well as on the used countermeasures on the device.
The following sections will give some insights in the way how to record the power traces and what has to be obeyed during recording. Also a quick excursion to statistics will be made to get some fundamental facts about the statistical properties of power traces which are needed in order to perform this type of attack.

5.2.1. Recording the Traces

Oscilloscopes can typically only measure voltage but not current. In order to measure the electric power a device consumes with an oscilloscope a small resistor is placed in the ground line. The voltage drop across this resistor can be measured and this measured value is proportional to the power consumption. A typical measurement setup can be seen in Figure 5.1.
Sometimes it is necessary to place the measuring resistor R_M in the V_{dd} line and not in the GND line. One reason for that could be that the device has more V_{dd} lines, one e.g. for the I/O-part ($V_{dd\_I/O}$), one for the encryption part (V_{dd_enc}) and so on. An attacker is of course only interested in the power consumption during encryption or decryption and so the measuring resistor will be placed in the appropriate V_{dd} line. It has to be obeyed that in

such a case a differential probe has to be used. Using a normal probe an oscilloscope always measures the voltage towards ground. A corresponding measurement setup can be seen in Fig. 5.2.

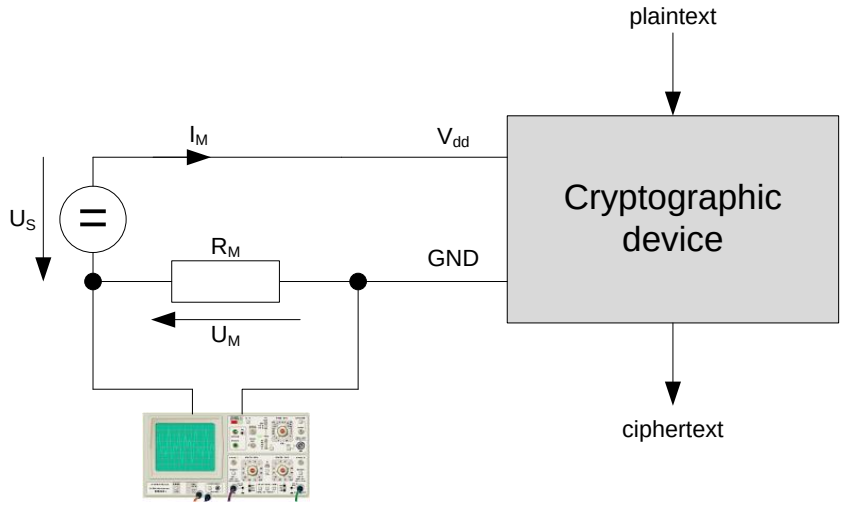

Figure 5.1.: Schematic overview of a typical power measurement setup.

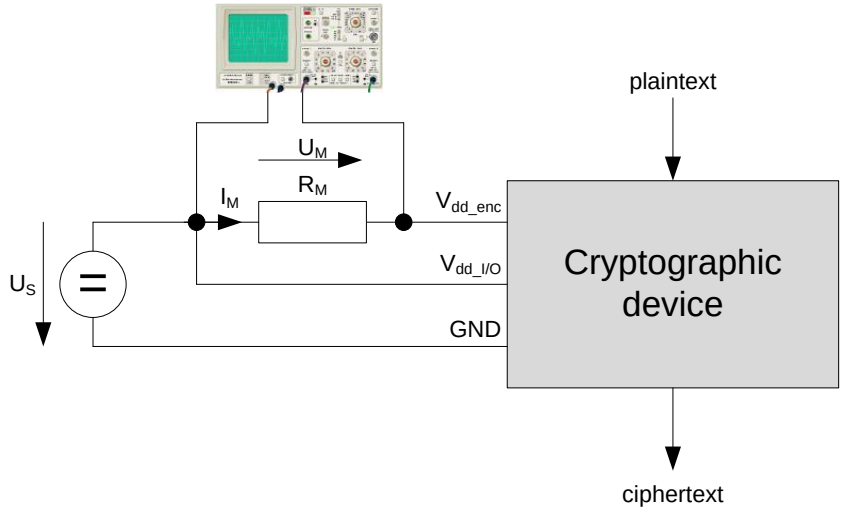

Figure 5.2.: Schematic overview of a power measurement setup with different V_{dd} lines.

Another possibility to measure the power consumption of a cryptographic device is using an EM-probe. In that case the electromagnetic emanation of the device or a single chip is measured. This emanation is proportional to the power consumption of the device. An advantage of this measurement setup is that no modifications in the circuit have to be done. The probe is placed on an adequate position close to the device or chip and no resistor has to be mounted inside the power or ground line. One disadvantage is that finding a good position for the probe might be quite challenging for some devices. Figure 5.3 shows

a single power trace of the CRYPTA chip for one AES encryption using the method with a measurement resistor in the power line. A power trace recorded using an EM-probe can be seen in Figure 5.4. Here an FPGA-chip was the attacked device running a CRYPTA-tag implementation. The trace was also recorded during an AES encryption.

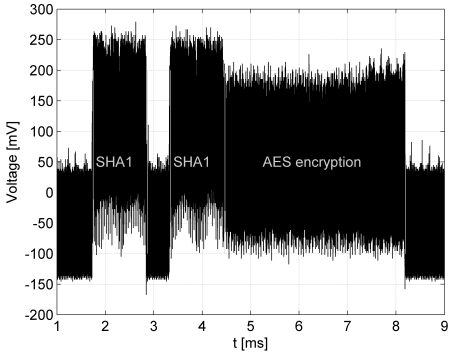

Figure 5.3.: Power trace of the CRYPTA tag during an AES encryption using a resistor in the GND-line.

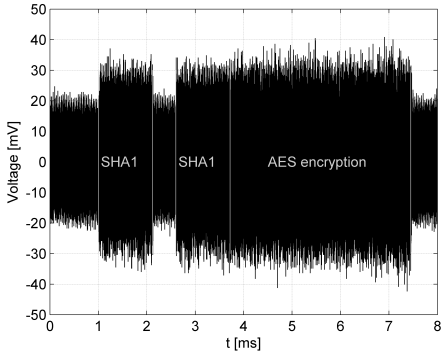

Figure 5.4.: EM trace of the FPGA prototype during an AES encryption.

An important factor when recording power traces especially for a DPA attack is the trigger signal. The better the trigger signal is the better the traces are aligned. Traces are perfectly aligned if the same operation (e.g. an S-box lookup for AES) occurs in all the recorded traces at exactly the same point in time. In most cases the traces are not perfectly aligned because it is typically hard to get a really good trigger signal for real-world cryptographic devices. So the recorded set of traces has to be aligned later on using e.g. a pattern-matching method. More detailed information about aligning methods can be found in [SM07] in chapter 8.2.2.

Another important factor is the frequency range of the traces. It is often useful to filter the traces in order to get rid of unwanted frequencies. One example is a power measurement of an RFID tag, where the carrier frequency is 13.54 MHz. It is very likely that the power traces are overlayed with that carrier frequency. Figure 5.5 shows such an overlayed trace before and after it has been filtered with a lowpass filter with a cut-off frequency of 13 MHz. Of course also other filter types like a bandpass filter or a bandstop filter may be suitable.

5.2.2. Statistical Properties

Next the focus will be on statistical properties of power traces. This part is mainly a summary of chapter 4 of the book [SM07]. Having a basic knowledge about the properties is a prerequisite to perform successful DPA attacks on cryptographic devices.

A power trace is a vector of values, where the length of the vector depends on the sampling rate of the oscilloscope on the one hand and on the recorded time of the trace on the other hand. Equation 5.1 shows the formula to calculate the length of the traces (*pts* denotes the

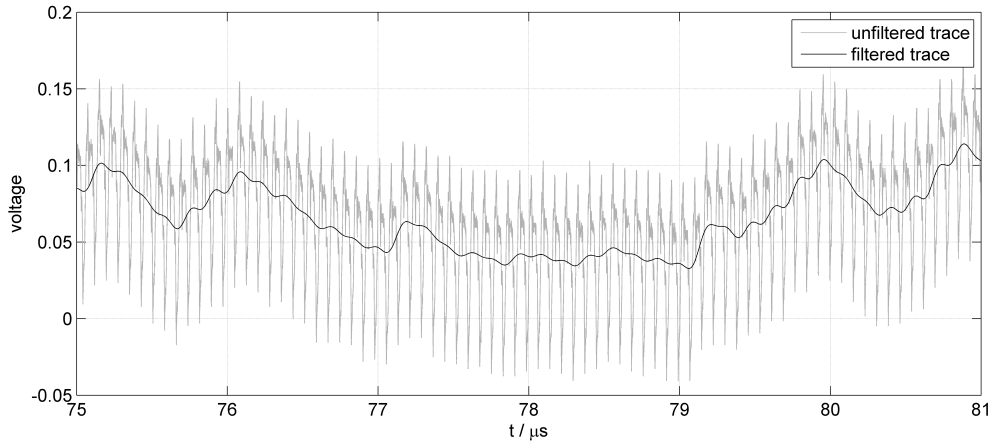

Figure 5.5.: Power trace before and after applying a lowpass filter.

number of points of the trace) where f_s is the sampling frequency in samples per seconds and t is the recorded time in seconds.

$$pts = f_s * t \tag{5.1}$$

It is important to keep the number of recorded points small to keep the amount of memory needed to store the traces small. Also the needed computational power to perform the attack increases with increasing length of the traces. Some ways to compress the traces will be discussed later in this chapter.
For example if the traces are recorded with $f_s = 2\,GS/s$ and the recorded time is $t = 50\,\mu s$, the power traces have a length of 100 000 points. If every point is stored as a double value with 8 bytes, each trace has a size of about 800 kB. For DPA attacks often 100 000 or even more traces are needed to reveal the key, so the memory requirement is quite high and the attacker will have to find ways to compress the traces to shrink their size.
In order to know which parts of a recorded power trace are important the factors which influence the power consumption have to be identified. The power consumption of a device at every point in time can be divided in the following parts:

- **Operation-dependent power consumption $\mathbf{P_{op}}$** This part is very important for power-analysis attacks as it describes the different amount of power a device needs for different operations.
- **Data-dependent power consumption $\mathbf{P_{data}}$** This is the second important part which describes how the power consumption changes because of the processed data. For different data that is processed, the device needs different amount of power.
- **Electronic noise $\mathbf{P_{el.noise}}$** This part is an unwanted part as it is the noise recorded during a measurement. For a single moment in time the noise can be reduced by recording the same power trace (device processes the same data and the same operation) several times and calculate the mean value.

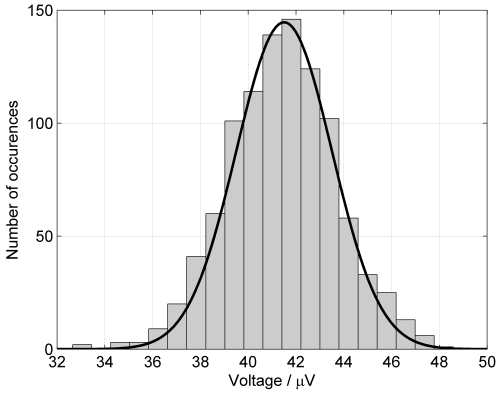

Figure 5.6.: Histogram of the voltage values.

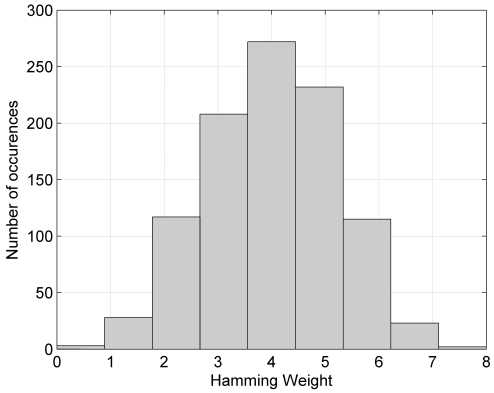

Figure 5.7.: Histogram of the data values after substitution.

- **Constant power consumption $\mathbf{P_{const}}$** This part also does not provide any information for an attack. One reason behind the constant part of the power consumption are leakage currents on the chip.

So each point of the power trace can be seen as the sum of the four parts mentioned above (Equation 5.2).

$$P_{total} = P_{op} + P_{data} + P_{el.noise} + P_{const} \tag{5.2}$$

It is important for power-analysis attacks to understand the influences of the unwanted parts $P_{el.noise}$ and P_{const} on the attack. There are some ways to minimize the influence of these parts. In order to get rid of the constant part of the power trace one could calculate the mean value of the whole trace and subtract this value from every point. So the trace becomes a zero-mean trace for further processing.
The electronic noise in every point of a power trace has a zero-mean gaussian noise distribution. In order to show this one has to find a data dependent point of the power trace and to record a large amount of power traces (e.g. 100 000) with constant data. Also the same operation has to be carried out at this fixed point for all the traces. Because of that assumptions the variance of P_{data} and P_{op} is zero. Next a histogram of the power values of all the traces at the fixed point of time can be drawn and the shape of this histogram looks like a normal distribution. The reason for this variation is the electronic noise.
Next the focus is put on the two parts influencing the power consumption which leak information on either the data processed (P_{data}) or the operation performed (P_{op}) at a fixed point of the power trace.
First we want to discuss the data dependency. The same experiment as with the electronic noise can also be used here, only the input data changes now. For the following example the FPGA implementation of the CRYPTA chip has been used. 1000 traces were recorded during the first round of AES. The input data were random values, which were stored to know the values for later analysis. As the point in time where the first byte is processed is known,

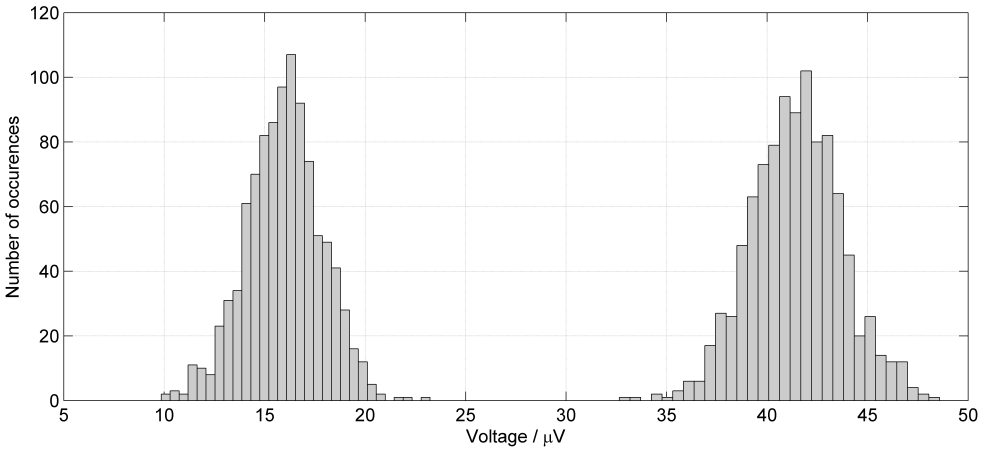

Figure 5.8.: Histogram of the voltage values for two different operations.

the voltage value of this point of time (exactly: the maximum of the trace in an interval) was extracted for all the 1000 traces and the histogram shown in Figure 5.6 was drawn. As mentioned in the attack section of Chapter 3, the value after the first substitution is attacked. So this value was calculated and a histogram of the hamming weight of the result could be plotted which can be found in Figure 5.7. When comparing the Figure 5.6 and Figure 5.7 one can see that both histograms look like a gaussian normal distribution. In Figure 5.6 the probability density function of a gaussian normal distribution with the mean value $\mu = 41.5\mu$V and a standard deviation $\sigma = 2\mu$V is also plotted to show this relation. From this fact an attacker can deduce that the hamming weight model is a good model for an attack on this chip!

The operation-dependent part of a power trace can be analyzed just like the data dependent part. The only difference now is that parts of the traces have to be found where different operations on the same data are performed. In many cases the different operations on the same data just lead to a different voltage level but the shape of the histogram stays the same. Figure 5.8 shows two histograms, where different operations were performed. It can be seen that the shape is nearly the same, just the mean value is different as the chip consumes different amount of power for different operations.

Another important statistical tool is the correlation. It describes the statistical relationship between two or more statistical variables, e.g. two different points on a power trace. The correlation is also used in DPA attacks to analyze the relationship between the power model and the real traces. This is described in more detail in Section 5.4. This section will just have a look on how to calculate the correlation in general. Equation 5.4 shows how to calculate the correlation coefficient ρ between two statistical variables X and Y using the covariance (Equation 5.3) and the variances.

$$Cov(X, Y) = E(X * Y) - E(X) * E(Y) \tag{5.3}$$

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X) * VAR(Y)}} \tag{5.4}$$

If one wants to calculate the correlation between two points of a trace the estimator r for the correlation coefficient can be used. Equation 5.5 shows how this estimator is calculated. Here it is assumed that there are two sets of traces X and Y. Every set contains at least n traces and $\overline{x}$ and $\overline{y}$ are the mean values of set X and Y.

$$r = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) * (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 * \sum_{i=1}^{n} (y_i - \bar{y})^2}} \tag{5.5}$$

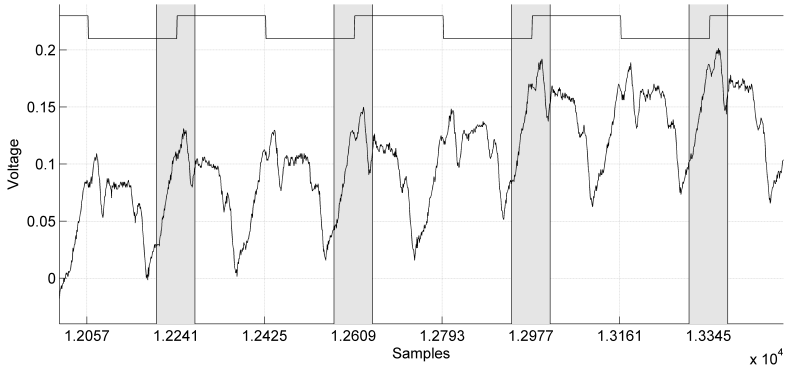

Figure 5.9.: The relevant parts of a power trace around the positive clock edge.

As mentioned at the beginning of this chapter the compression of the power traces is an essential part of a power-analysis attack to decrease memory requirements and computation time. So the relevant parts of the power traces have to be found and the rest of the data can be ignored for analysis.
Maximum extraction is one possibility to shrink the size of the power traces. In this special case only the maximum value of the power trace in a defined interval is used. For example the size of the power trace equals 1000 samples and the length of the interval is 100 samples. Here the compressed trace consists of 10 points, the maximum values of the intervals: 1 to 100, 101 to 200, ... 901 to 1000.
Another way to compress the recorded traces is integration. Here a defined number of consecutive points of the trace are summed up. There are different ways how this summation can be done. The raw values can be summed up, the sum of absolute values or the sum of squares can be calculated.
For many attacks only the power consumption at the rising or the falling clock edge is relevant. So only the parts of the power traces around these points in time have to be considered for the analysis. This fact can be seen in Figure 5.9. The power trace in this figure is from the CRYPTA chip during an AES encryption.

5.3. Simple Power Analysis

Simple power analysis (SPA) means that one only needs a few power traces to get the wanted value, e.g. a part of the secret key. The attacker needs quite a lot of knowledge about the target device to be able to perform an SPA attack. If this knowledge is present, this attack is really mighty as only a small number of traces is needed.
The key to a successful SPA attack is the knowledge, at which point in time of the power trace which instruction is performed and the fact that the power consumption of this instruction depends on the operands. If one of these operands is a part of the secret key it can be revealed with an SPA attack. This can be done by inspection of just one power trace. Figure 5.10 shows a part of the power trace of the CRYPTA chip. In this section of the trace the first AES round on the first byte of the plaintext is performed. This part only consists of the key addition and an S-box lookup. It can be seen that the amplitude of the traces depend on the Hamming weights of the output of the S-box. For small Hamming weights the power consumption is low and for higher Hamming weights it is higher.

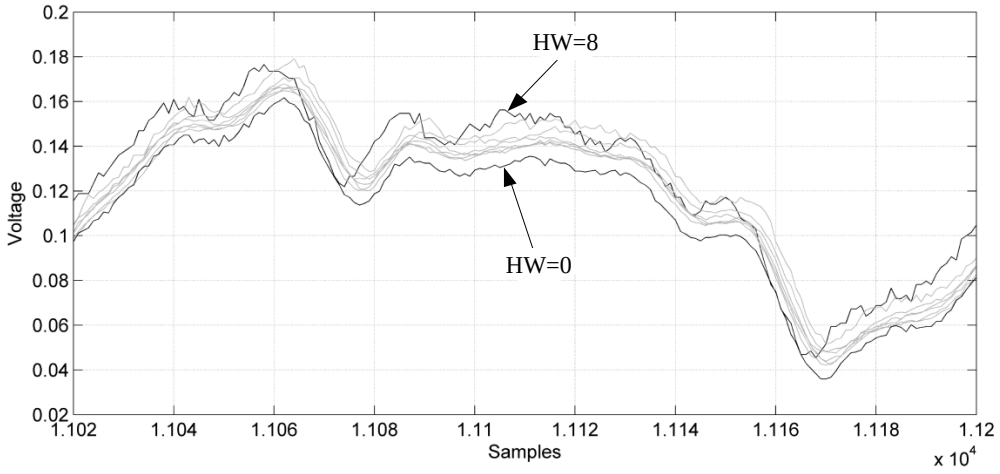

Figure 5.10.: Zoomed view of power traces with different Hamming weight of processed data.

Another way to perform an SPA attack is called template attack. In this kind of attack the attacker possesses a similar device as the target device and it consists of two phases: In the first phase, the template-building phase, templates are created using the similar device. These templates are also power traces where the input data and the key is known. In the second phase, the template matching phase, the power trace of the target device is compared with the previously-recorded templates. Using statistical methods the template which matches best can be found. The key that was used to create the best-matching template is known. So it is rather likely that the target device used the same key. In [Med07] a template-based SPA attack on an ECDSA implementation is described and in [Osw03] an enhanced SPA attack on ECDSA is presented.

5.4. Differential Power Analysis

Performing a DPA attack on a cryptographic device requires a large set of power traces in contrast to SPA. It is assumed that the attacker physically possesses the device for a couple of time. During this time a large amount of chosen plaintexts can be encrypted and the corresponding power traces can be recorded. Of course also the other direction, where the attacked device decrypts chosen ciphertexts, is possible. The advantage of a DPA attack is that the adversary does not need a lot of knowledge about how the cryptographic algorithm is implemented. It is even enough to know the type of the cryptographic algorithm which is used in many cases.

5.4.1. The 5 Steps of a DPA Attack

The sequence of a DPA attack is quite similar for all attacks and consists of 5 steps (see Figure 5.14 for additional information):

- **Step 1: Choosing an intermediate result** In the first step an intermediate result v of the implemented algorithm has to be found which is suitable for the attack. This result should be a function of some known data d and the secret key k ($v = f(d, k)$). See 3.4 for two examples for such intermediate results of AES.
- **Step 2: Recording the power traces** Here one has to record the power traces while the device processes the known data d. The number of traces which is recorded is D and so the known data can be written in vector form: $d=[d_1,\ d_2\ \ldots\ d_D]$. The number of samples of each trace is denoted with T. The traces are stored in a matrix $\mathbf{T}$ with D rows and T columns.
- **Step 3: Calculating the intermediate results** After the traces are recorded the intermediate results $v_{i,j}$ which were chosen in step 1 are calculated. In this step also the key hypothesis k has to be chosen. The number of hypotheses is denoted with K. So k can be written as vector as well: $k=[k_1,\ k_2\ \ldots\ k_K]$. The result of this step is a matrix $\mathbf{V}$ with the values: $v_{i,j}=f(d_i,k_j)$. So the matrix $\mathbf{V}$ has D rows and K columns.
- **Step 4: Finding a power model** In the fourth step the intermediate values are mapped to hypothetical power values using a power model. Finding a suitable power model is the easier, the better the implementation of the algorithm is known. An example for a power model is the Hamming-weight model. Here the assumption is that the higher the number of ones in the intermediate result is the higher is the power consumption. The result of this step is a matrix $\mathbf{H}$ which has D rows and K columns.
- **Step 5: Comparing the hypothetical power values with the recorded power traces** In the last step the hypothetical power values are compared with the real power traces. For this purpose each column of the matrix $\mathbf{H}$ is compared with each column of the matrix $\mathbf{T}$. The result is a matrix $\mathbf{R}$. This matrix has K rows and T columns. The value $r_{i,j}$ denotes how good column i of matrix $\mathbf{H}$ fits to column j of power-trace matrix

T. The correlation coefficient can be used to calculate the values of the matrix **R**. The following section describes how to calculate the correlation values.

5.4.2. DPA Attack Based on the Correlation Coefficient

For comparing different sets of data the correlation coefficient is often used. The correlation coefficient has been discussed earlier in this chapter for comparing different points in time of a power trace (see Equation 5.5). Now the correlation between the hypothetical power values (columns of **H**) and the real power traces (columns of **T**) has to be calculated. Equation 5.6 shows how to calculate the correlation coefficient between the two columns. $\overline{h_i}$ denotes the mean value of column i of the matrix **H** and $\overline{t_j}$ the mean value of column j of matrix **T**, respectively.

$$r_{i,j} = \frac{\sum_{n=1}^{D} (h_{n,i} - \bar{h_i}) * (t_{n,j} - \bar{t_j})}{\sqrt{\sum_{n=1}^{D} (h_{n,i} - \bar{h_i})^2 * \sum_{n=1}^{D} (t_{n,j} - \bar{t_j})^2}} \tag{5.6}$$

The matrix **R** is typically visualized by plotting the rows as a graph. The result of a DPA attack on the the CRYPTA tag can be viewed in Figure 5.11. The black graph (correct hypothesis) is the graph with the highest correlation value (equals the row where the highest correlation value appears) and the other rows (wrong key hypotheses) are plotted in gray. The highest peak in the graph is at about 80μs so it can be said that at this point in time the chosen intermediate result is calculated.

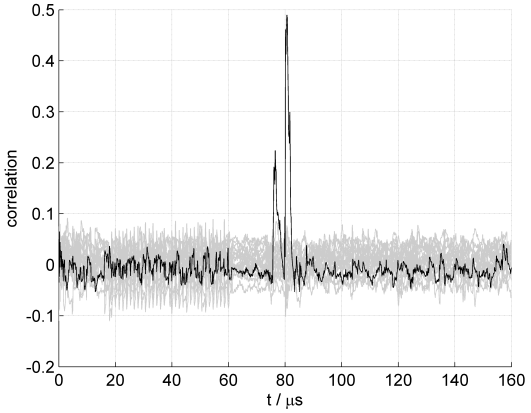

Figure 5.11.: Result of a DPA attack on the CRYPTA tag.

The more traces are recorded the higher is the probability that the result of the DPA attack is correct. Also the effort for the attack increases with the number of power traces so it is of great importance to know how much traces have to be recorded approximately to get a correct result with a high probability.

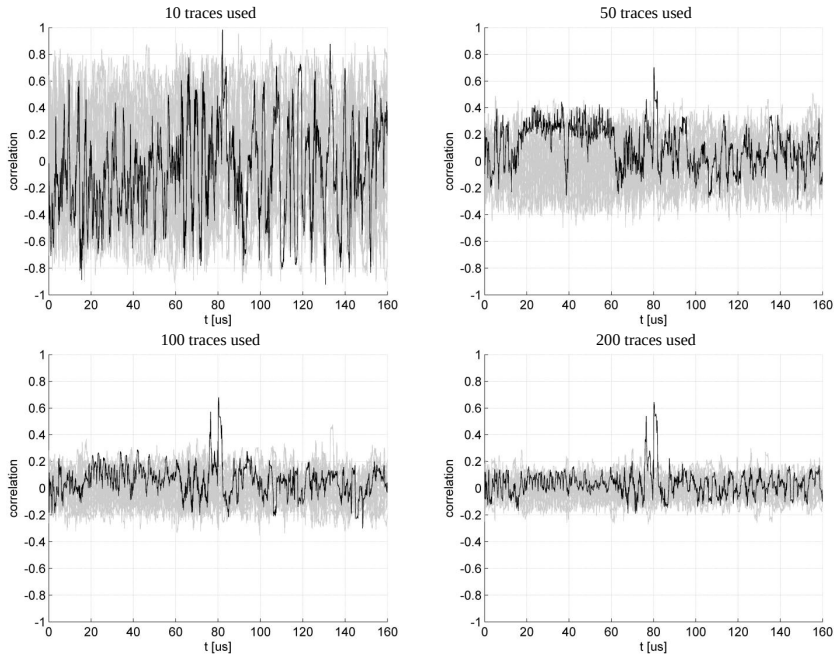

Figure 5.12.: Results of DPA attacks on the CRYPTA tag that use different numbers of traces.

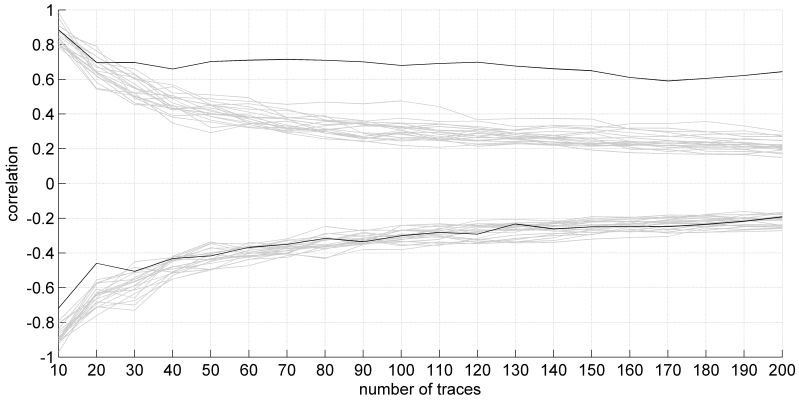

Figure 5.13.: Evolution of the correlation coefficients.

Figure 5.12 shows several results of DPA attacks on the CRYPTA tag. The black trace relates to the correct key hypothesis (highest correlation value) and the gray traces relate to the wrong key hypotheses. It can be seen that the amplitudes of the gray traces decrease with increasing number of traces while the maximum value of the black trace, which belongs to the correct key hypothesis does not depend on the number of traces. The evolution based on the number of traces of the correlation coefficients of wrong and correct key hypotheses can

also be seen in Figure 5.13.
Mangard et al. [SM07] present a simple equation for estimating the maximum amplitude of the correlation coefficient of the wrong key hypotheses (here called *noisefloor*). It only depends on the number of traces n. So if one knows what the maximum expected correlation coefficient in the attack for the correct key hypothesis will be the number of traces needed for a successful attack can be calculated using Equation 5.7. For a small number of traces (less than 1 000) this equation is not that accurate so it should be used carefully there.

$$noisefloor = \pm\frac{4}{\sqrt{n}} \tag{5.7}$$

n	noisefloor
50	0.57
100	0.40
200	0.28
1000	0.13
10000	0.04

Table 5.1.: Values for the *noisefloor* for different numbers of traces n.

Comparing the calculated values from Table 5.1 with the values from the DPA attack in Figure 5.12 and Figure 5.13 shows that the equation matches the real results quite well although the number of used traces is quite low!
The way how to derive this solution can be found in [SM07] as it would go beyond the scope of this work.

5.4.3. Related Work on DPA

Differential power analysis was introduced in 1999 by Kocher et al. [KJJ99]. The authors describe an attack on the DES using difference of means.
The book [SM07] provides insights in all the fields of power analysis attacks. In this book attacks on an AES implementation on a microcontroller as well as on an FPGA are presented.
In the work of Fischer et al. [FGKV07] DPA attacks on different stream ciphers are presented.
A way to enhance a DPA attack using the difference-of-means technique is presented in [BK03]. Using the presented approach one can decrease the complexity of the attack by a factor of two.
In [Mes00] an approach for a second-order DPA attack is presented which can reveal a secret key where a first-order DPA attack is not successful.
Joye et al. [JPS05] also analyze higher-order DPA attacks in their work.
Hutter et al. [HMF07] present power and EM attacks on 13.56 MHz RFID devices in their paper. The attacks target an unprotected AES implementation.

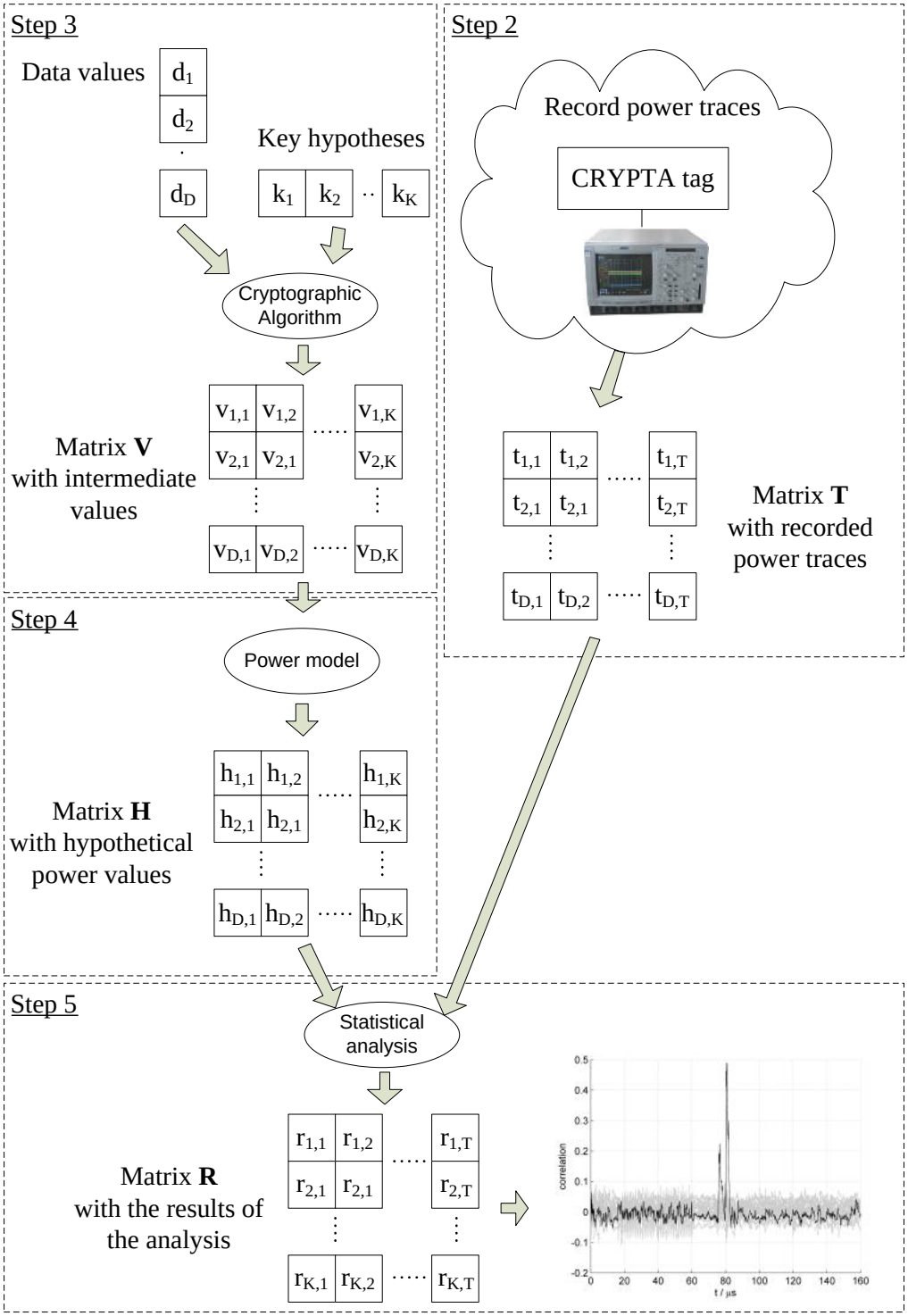

Figure 5.14.: The sequence of a DPA attack.

5.5. Countermeasures Against Power-Analysis Attacks

In this section we want to present some countermeasures to prevent power-analysis attacks. Mangard et al. [SM07] split the countermeasures against power-analysis attacks into two groups: hiding countermeasures and masking. **Dummy operations** and **shuffling** belong to the group of hiding countermeasures and they are also used on the CRYPTA tag.

5.5.1. Dummy Operations

The insertion of dummy operations increases the complexity of power analysis attacks significantly. The idea here is to execute an instruction at different points in time for different runs. If three dummy operations are used the real instruction can be executed at four different points in time. It must not be possible to distinguish between a dummy operation and the real instruction and the insertion of the dummy operations must be randomized. As it is essential for an attacker performing a DPA attack to know at which point in the power trace the intermediate result is calculated the randomization increases the complexity of the attack. As a result more power traces are needed to perform a successful attack.
In case of AES the dummy operations can be extended to dummy rounds. In this case rounds performing the round transformations on a dummy state are inserted. If the attacker is targeting a specific round for his attack the probability that this round works on the real state decreases with the number of possible dummy rounds. For DPA attacks an intermediate result has to be calculated which depends on some known data and a part of the key. As the known data is the plaintext or the ciphertext most of the time the attacker will attack one of the first two or last two rounds of AES. So it is sufficient to place dummy rounds before the first round, between first and second round and before and after the last round. Of course the number of dummy rounds placed at each position has to be randomized again.
One disadvantage of the insertion of dummy rounds/operations is that the runtime of the algorithm increases. Here a trade-off between performance and security has to be found.
More information on this countermeasure can be found in [SM07]. One way how to implement dummy rounds to secure AES is presented in [HFW11].

5.5.2. Shuffling

Shuffling means to change the sequence of operations of an algorithm in the time domain. In this case the runtime of the algorithm does not change but the implementation might be more complex.
In case of AES shuffling could randomize the sequence of processing the bytes of the state. In DPA attacks the attacker focuses on one specific byte of the state. As the processing of this byte can appear at 16 different positions in the power trace the correlation coefficient of the correct key hypothesis decreases and so more traces are needed in order to get a correct result.

5.5.3. Masking

In the case of masking the intermediate values are concatenated with a random mask m. For every new run of the algorithm a new random mask is generated. The mask has to be random to make it hard for an attacker to find the value of the used mask. As the intermediate values are masked the power consumption is no more predictable by the attacker and so the standard DPA attack will not succeed. One way to secure AES using masking is presented in [CHM06].

6. Functionality-Test Results

In this chapter we present the results of the functionality tests of the chip. As mentioned in the introduction it is very important to run a lot of tests on prototype chips to show that the design does not have errors or if faults are found to correct them.
The chapter is organized as follows. First, an overview of the test setup is given. We continue with the results for the file operations (reading files and writing files) followed by the test results for the cryptographic unit. Then the results for the different data rates for communication between reader and tag are presented. At the end of the chapter a short summary of the functionality-test results can be found.

6.1. Test Setup

In order to test the functionality of the CRYPTA chip, three different readers were used: the IAIK HF reader, the TAGNOLOGY TAGSCAN reader and the Nokia 6212 NFC-enabled mobile phone.
With the IAIK HF reader and the TAGNOLOGY TAGSCAN reader all the functions of the tag could be tested. For controlling the readers an existing software framework (IRIS) developed in Java has been used.
Using the mobile phone only a subset of the functions could be tested because an existing application for communication between mobile phone and tag has been used. More information on this application can be found in [Kor10]. The mobile-phone application supports reader and tag authentication using the Advanced Encryption Standard (AES), reading and writing the NFC data exchange format (NDEF) file and verifying signatures from the tag using the Elliptic Curve Digital Signature Algorithm (ECDSA). It is not possible to try to read out the AES secret key with the mobile phone as an example. For this purpose a new software would have to be developed that would be out of the scope of this work. Tests like this can be performed using one of the other two readers. The main target of using the mobile phone as a reader was to show how the tag might be used in everyday life. Consumers will use a mobile phone like the Nokia 6212 to communicate with the tag.
The tests in Section 6.2 were performed with the IAIK HF reader and the TAGSCAN reader. In order to indicate the status the tag uses status words in the end of the answer to a command. All the supported status words as well as their meaning can be found in Table 2.7.

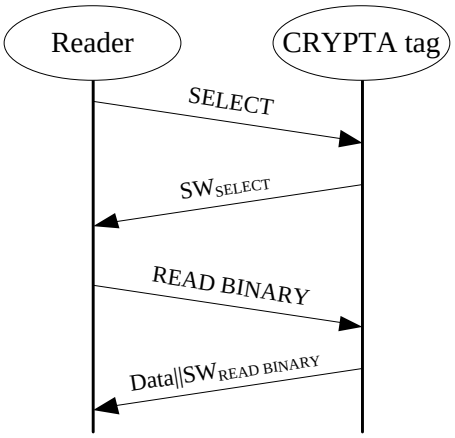

Figure 6.1.: Communication sequence for reading a file.

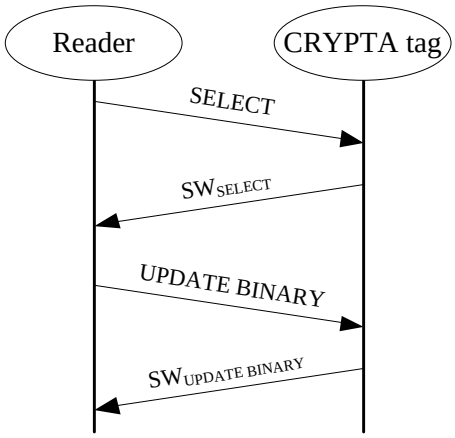

Figure 6.2.: Communication sequence for writing a file.

6.2. File Operations

For reading files the most important thing was to test if the protection of the secret-key files is ensured. The next thing to test here was the correct length of the files to read. It must not be possible to read e.g. 10 bytes if the file length is only 8 bytes. In this case also the correct status word could be checked. With the configuration byte 1 (CFG1) it is possible to allow a read operation only after a succesful external authentication. For this purpose bit 7 of this byte has to be set. Figure 6.1 shows the command sequence for a reading operation. With the status word SW_{SELECT} and $SW_{READ\ BINARY}$ the tag indicates whether the SELECT and READ BINARY commands were successful or not.
Also many different protection mechanisms to prevent the unauthorized modification of the information stored in the EEPROM are implemented on the CRYPTA chip. Some files can only be written during production or during personalization where other files can only be written once (one-time programmable). For some pages of the EEPROM a lock bit indicates if the user can write to these pages during normal operation. If bit 6 of CFG1 is set any writing operation is only allowed after an external authentication. For more information on the different file-protection mechanisms see Section 2.6. Next, the test procedure for the above mentioned file operations are explained in detail and the results are shown. For the tests the personalization bit of the chip is set (i.e. the tag is personalized) if no other assumptions are mentioned. Figure 6.2 shows the command sequence for a write operation. With the status word SW_{SELECT} and $SW_{UPDATE\ BINARY}$ the tag indicates whether the SELECT and UPDATE BINARY commands were successful or not.

6.2.1. Read Access of Protected Files

As it can be seen in Table 2.1 the following files are read-protected: ECDSA private key, AES secret key, and true random number generator (TRNG) seed. In order to test the protection mechanism a SELECT command with the appropriate file identifier was send to the tag. The answer of the tag was the status word `0x6982`, which means that the select operation was not successful. The security status of the file prevents selecting it. The same results could be achieved for all the protected files. Therefore, the protection of the files works and the chip passed this test.

6.2.2. Correct Reading Length

The correct reading length was tested for all the readable files (all files except the protected ones). In this case the file to test was selected using the SELECT command with the appropriate file identifier. The chip returned the status word `0x9000` to indicate the successful selection of the file. Next, the READ BINARY command was send to the tag, but with a length parameter exceeding the real length of the file. Now the answer of the tag was the data of the file followed by the status word `0x6282`. This status word indicates that the end of the file was reached before the expected number of bytes could be read. This result was the same for all the files and so the chip passed this test.

6.2.3. Read Access When External Authentication is Required

As mentioned at the beginning of this section the reading operation can be protected with an authentication beforehand if the 7^{th} bit of CFG1 is set. For testing this functionality normal read operations were performed on the different files, once without an external authentication and once with an external authentication beforehand. If an external authentication was performed beforehand, the tag always sent the correct data followed by the status word `0x9000` indicating that the read operation was successful. Without an external authentication the answer of the tag on the READ BINARY command was always `0x6982`, which means that the security status prevents the reading operation. For all the file IDs the same result could be achieved and so the chip passed this test.

6.2.4. Invalid File IDs

If an unknown file ID is used the chip has to signalize this with a status word. In order to test this the SELECT command with an invalid file ID was sent to the tag. This was done several times with random IDs. The answer of the tag to the SELECT command with a wrong file ID always was the status word `0x6A82`. This status word indicates that the file ID does not exist. As the file ID is just a 16-bit value all the IDs could be covered with one test run and the answer of the tag was always correct, so the chip also passed this test.

6.2.5. Write Protection

As it can be seen in table Table 2.1 writing to the following files must not be possible: public-key certificate, ECDSA private key, AES secret key, UID, CFG1, CFG2, page RW status and TRNG seed. The test procedure here was the following: The file to test was selected using the SELECT command with the appropriate file ID. For files where reading is not allowed (ECDSA private key, AES secret key, TRNG seed) the answer of the tag on the SELECT command was `0x6982` indicating that these files must not be selected because of the protection mechanism. For the other files the answer of the tag was `0x9000` indicating that the selection was successful. Next, an UPDATE BINARY command (in standard write mode) was send to the tag. For all the files the answer of the tag to the UPDATE BINARY command was `0x6982` indicating that the security status prevents writing. For the UPDATE BINARY command in personalization mode the same results could be achieved. As all the answers of the tag were correct it passed this test.

6.2.6. Writing VALUE1 File

The file VALUE1 with file ID `0x0007` has a special protection. The content of this file can only be updated if the new value v_{new} is smaller than the previous one v_{cur}. In order to test this functionality first the file with ID `0x0007` was selected. Next the current value v_{cur} stored on this position was read. Using the UPDATE BINARY command a value v_{new} was send to the tag. For values satisfying $v_{new} > v_{cur}$ the answer of the tag was `0x6982` which means that the update binary operation was not successful. If $v_{new} == v_{cur}$ the answer of the tag was also `0x6982`. Only for values satisfying $v_{new} < v_{cur}$ the answer of the tag was `0x9000` indicating that the update binary command was successful.

6.2.7. Testing the Lock Bits

As mentioned earlier in this work the 24 bits of the page RW status indicate whether write operations to the pages `0x08` to `0x1F` are allowed or not. If e.g. only the first bit is set writing is only allowed on page `0x08`. Using this mechanism writing on the NDEF file, the CC file and the user memory can be allowed or not. In order to test this mechanism different values for the page RW status were used and then writing operations on the affected pages were performed. If writing on the page was allowed the answer of the tag to the UPDATE BINARY command was `0x9000` indicating that the write operation was successful. On the other hand if writing on this page was not allowed the answer of the tag was `0x6982` which means that the writing operation was not successful. As this is the expected behavior, the chip passed this test.

6.2.8. Writing Only After External Authentication

As mentioned at the beginning of this section, bit 6 of CFG1 can be used to select whether writing the files user memory, NDEF file, CC file and VALUE1 is only allowed after a successful

authentication of the reader. All bits of the page RW status were set so as to test this security feature. With this setting writing to all pages is allowed. Using the SELECT command with the appropriate file id one of the four files previously mentioned was selected. If the status word of the tag was `0x9000` (selection successful) the UPDATE BINARY command was send to the tag. If an external authentication was performed beforehand the answer of the tag was `0x9000` (writing successful), otherwise the answer was `0x6982`. As we got similar results for all the files (VALUE1: $v_{new} < v_{cur}$ had to be satisfied) the chip passed this test.

6.2.9. Testing the Correct Write Length

As mentioned in Chapter 2 the maximum write length is 16 bytes. It is only possible to perform a write operation on one page of the EEPROM with one UPDATE BINARY command. Four files can be written during normal operation (CC file, NDEF file, user memory, VALUE1 file). There are two test scenarios to cover all the functionalities.
In the first scenario the focus is put on the VALUE1 file. The length of this file is 2 bytes and after this file the CFG2 byte follows on the same page of the EEPROM. This can be seen in Figure 2.3 showing the EEPROM layout of the CRYPTA chip. It must not be possible to write a three-byte long value into the VALUE1 file as this would overwrite the CFG2 file. In order to test this the VALUE1 file was selected using the SELECT command. The answer of the tag was `0x9000`. Next an UPDATE BINARY command was sent to the tag with a length parameter of three and a three-byte long data. The answer of the tag was `0x6700` indicating that the length of the data was wrong. As this was the expected behavior the chip passed this test.
The second test scenario deals with the three other previously mentioned files. Here, the fact that only one page can be written prevents from writing to another file than the selected one because of a wrong length parameter. So the only thing to test here was an UPDATE BINARY command with a length parameter exceeding 16. The answer of the tag to this command was `0x6700` indicating that the length parameter was wrong. As this was the expected answer the chip passed this test.

6.3. Cryptographic Unit

As mentioned in Chapter 2 the cryptographic unit of the chip can run with four different clock frequencies: 848 kHz, 1.7 MHz, 3.4 MHz and 6.8 MHz. The frequencies can be selected using the lower two bits of CFG1. Tests for the proper function of the AES and the ECDSA implementation are presented in this section.
As there was a different behavior for the different readers the results are presented for every reader separately. The three cryptographic operations that the tag supports are also divided into two groups: AES and SHA1 belong to the first group and ECDSA belongs to the second group. The need for this separation rises because the duration for an ECDSA calculation is much higher compared to the duration for an AES calculation or generating a challenge and so different behavior appeared.

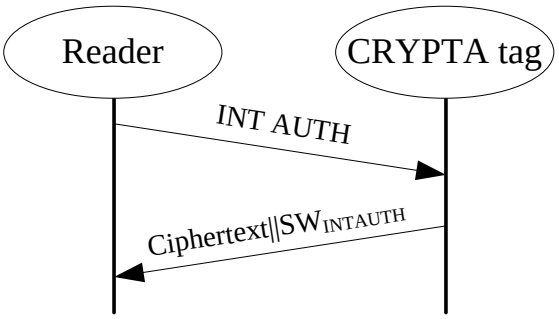

Figure 6.3.: Command sequence for a tag authentication.

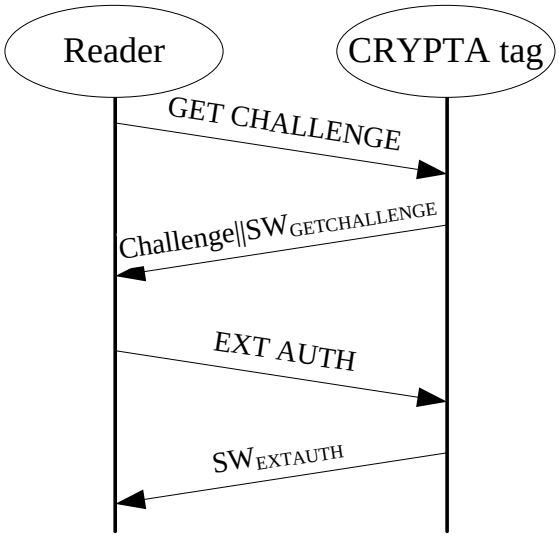

Figure 6.4.: Command sequence for a reader authentication.

6.3.1. Test Scenarios AES and SHA1

There are two commands that start an AES calculation on the chip and these commands are the following: INTERNAL AUTHENTICATE AES and EXTERNAL AUTHENTICATE. With the INTERNAL AUTHENTICATE AES command the tag authenticates to the reader. It encrypts a challenge received from the reader using AES. The tag then sends the ciphertext back to the reader followed by the status word `0x9000`. The correct encryption of the tag can be verified by decrypting the received ciphertext and comparing the result with the challenge that was sent. Figure 6.3 shows the command sequence for the internal authentication.
With the EXTERNAL AUTHENTICATE command the reader authenticates to the tag. In this case the reader has to send a GET CHALLENGE command to the tag beforehand to receive a challenge from the tag. The reader then encrypts this challenge using AES and sends it back to the tag. The tag verifies the ciphertext and indicates with a status word as answer if the authentication was successful. One bit in the challenge of the tag was flipped in order to test the status word of the tag for a failed authentication. Figure 6.4 shows the command sequence for an external authentication. Before the tag starts the AES encryption/decryption it requests a WTX with a value of `0x05`.
With these test scenarios the AES implementation on the tag could be tested and additionally the SHA1 implementation could be verified. SHA1 is used to generate a random number as part of the internal authentication process and the challenge generation process. These scenarios were performed for all the four frequencies of the cryptographic unit. During the tests we discovered a problem concerning the power supply of the chip. In some cases the value for the supply voltage fell below 1.6 V because of a too high power consumption. As a result the chip resets itself and does not finish the requested command. In order to prevent this an extra capacitor has been placed between the power line and the ground line of the chip located on the testboard. The values for this extra capacitor to pass the test can be

found in Table 6.1. The checkmark in the table denote that no capacitor was needed for the reader-frequency combination.

	Reader		
	TAGSCAN	IAIK HF reader	Nokia 6212 mobile phone
848 kHz	✓	2.80 nF	✓
1.7 MHz	✓	2.80 nF	✓
3.4 MHz	✓	4.48 nF	✓
6.8 MHz	220 nF	1.00 μF	✓

Table 6.1.: Test results for the AES implementation of the CRYPTA chip.

Results for the TAGSCAN reader
As it can be seen in Table 6.1 when using the TAGSCAN reader a capacitor is needed for the highest frequency of the cryptographic unit. With a higher frequency also the amount of power rises that the chip needs to perform the cryptographic operations. As the available power on an RFID device is limited the supply voltage decreases and if this voltage drop is too large the chip resets itself. So the capacitor is needed to support the supply voltage generated from the reader field.

Results for the IAIK HF reader
For a proper functionality with the IAIK HF reader an additional capacitance was needed for all the frequencies of the cryptographic unit. There is a difference in the field strength between the TAGSCAN reader and the IAIK HF reader and so the behavior is different. The fact that the power consumption at 6.8 MHz is much higher than for the lower frequencies can be seen in the step of the capacitor value from 3.4 MHz to 6.8 MHz. Figure 6.5 shows the recorded trace of the supply voltage without an extra capacitor (tag resets itself) and Figure 6.6 shows the recorded trace of the supply voltage with an extra capacitor (tag successfully performs cryptographic operation).

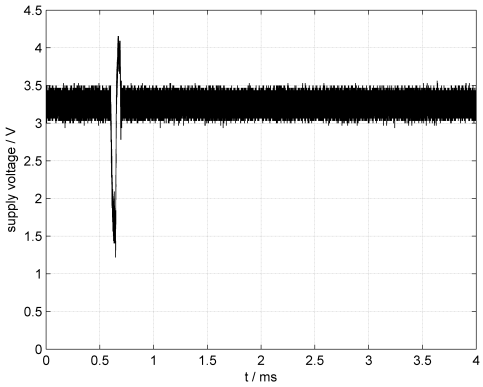

Figure 6.5.: Voltage drop without capacitor.

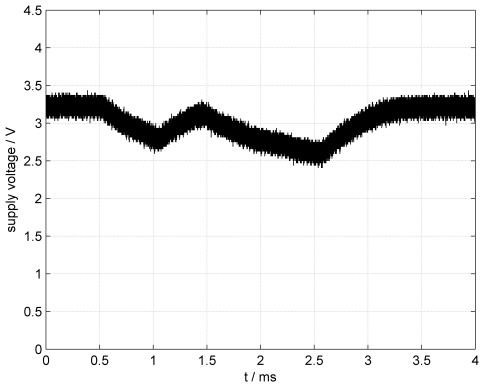

Figure 6.6.: Voltage drop with capacitor.

Results for the Nokia 6212 Mobile Phone
The internal and external authentication can also be performed with the Nokia 6212 mobile phone. As it can be seen in Table 6.1 AES encryption as well as generating a challenge worked with the mobile phone as a reader with all the frequencies of the cryptographic unit. No external capacitor was needed.

6.3.2. Test Scenarios ECDSA

In order to test the ECDSA implementation the INTERNAL AUTHENTICATE ECDSA command was used. The reader sends a 16-byte long challenge within this command to the tag and the tag signs this challenge using the ECDSA. The answer of the tag is then the signature which is 48-bytes long. At the reader this signature can then be verified as the public ECDSA key can be read from the tag memory. As generating the ECDSA signature is a time-consuming operation, the tag sends a waiting time extension (WTX) with the value `0x40` to the reader to indicate the delay.

	Reader		
	TAGSCAN	IAIK HF reader	Nokia 6212 mbile phone
848 kHz	✓	2.80 nF	x
1.7 MHz	✓	3.36 nF	x
3.4 MHz	✓	4.48 nF	x
6.8 MHz	47 μF	100 μF	x

Table 6.2.: Test results for the ECDSA implementation of the CRYPTA chip.

Results for the TAGSCAN reader
As it can be seen in Table 6.2, when using the TAGSCAN reader an extra capacitor is only needed when the highest frequency of the cryptographic unit is used. The only difference to AES is that the value is much bigger. The higher value is needed because the duration to generate the ECDSA signature is much longer in comparison to encrypt a challenge using AES. The longer the cryptographic unit is active the higher is the energy requirement of the chip. With this setup a valid signature could be received for all of the four clock frequencies.

Results for the IAIK HF reader
When using the IAIK HF reader an extra capacitor is again needed for all the frequencies of the cryptographic unit. It figures out that the values of the capacitor for 848 kHz, 1.7 MHz and 3.4 MHz are small compared to the value needed for 6.8 MHz. Only for the highest frequency the capacitor is needed to stabilize the supply voltage of the chip because of the high power consumption. For the three lower frequencies the power consumption is no problem but the capacitor supports the communication between reader and chip in that cases. Without a capacitor no proper communication could be established.
Figure 6.7 shows the trace of the supply voltage for a frequency of 1.7 MHz of the cryptographic unit. The duration for generating a signature is about 480 ms. Figure 6.8 shows the supply-voltage trace for a frequency of 3.4 MHz of the cryptographic unit. Now, the duration

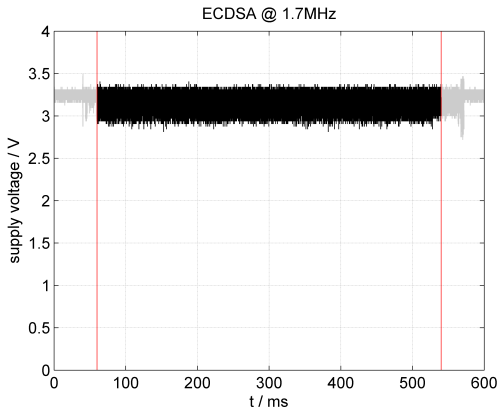

Figure 6.7.: Supply voltage of the CRYPTA tag during ECDSA signature generation (1.7 MHz).

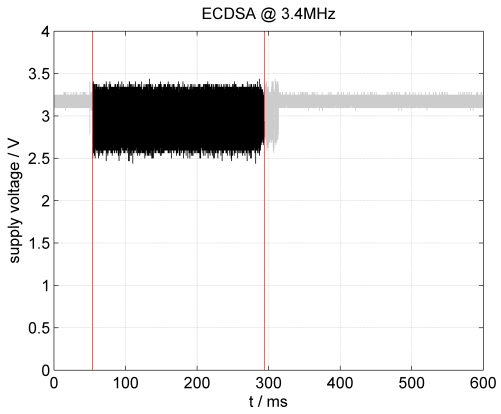

Figure 6.8.: Supply voltage of the CRYPTA tag during ECDSA signature generation (3.4 MHz).

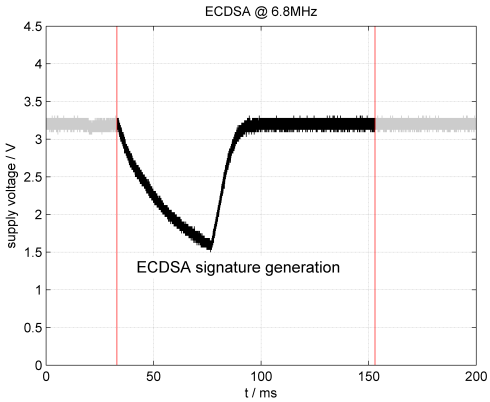

Figure 6.9.: Capacitor with too small value (10 μF).

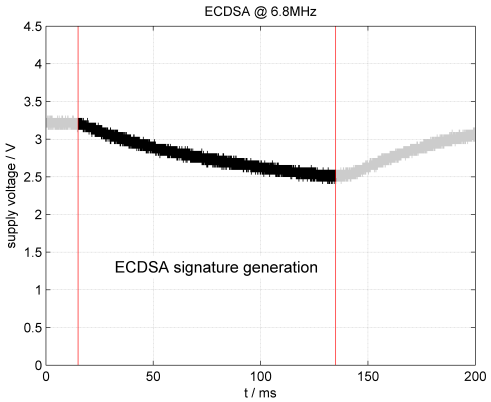

Figure 6.10.: Capacitor with appropriate value (100 μF).

for generating the signature has halved to 240 ms compared to Figure 6.7 because of the higher frequency. Figure 6.9 and Figure 6.10 show the trace of the supply voltage during a signature generation with a clock frequency of 6.8 MHz. In both cases a capacitor was used but in Figure 6.9 the value of the used capacitor was too small. The result of this too small value is that the supply voltage decreases and reaches the value where the tag resets itself (about 1.6V) before the signature generation is finished. In Figure 6.10 the value of the used capacitor was big enough for finishing the signature generation.

Results for Nokia 6212 Mobile Phone

As the application on the Nokia mobile phone also supports tag authentication using ECDSA the tests were also performed using this device. The test cases with this reader did not produce any positive results.

The debug output on the mobile phone was used to identify the reason for this incorrect operation. With this extra information an incorrect status word from the tag as answer to the INTERNAL AUTHENTICATE ECDSA command could be found. Instead of the status word `0x9000` the tag sent the status word `0x6d00` (command not supported or invalid). In order to refine the error search the IAIK DemoTag was used. With this device it is possible to intercept a communication between reader and tag. We found out that the answer of the reader to the WTX of the tag was wrong. A correct answer of a reader to a WTX is a WTX command with the same value as the received one. In our case the reader received a WTX with the value `0x40` but sent a WTX with `0x00`. After some research it figured out that only the lower six bits of the WTX value can be used [Int08]. The mobile phone sets the upper two bits of the WTX value to 0 and so the value `0x40` becomes `0x00`. As the tag expects a WTX with `0x40` but receives `0x00` it does not start the ECDSA signature generation and indicates this with the status word `0x6d00`. It is important to mention that the mobile phone works according to the standard but the tag does not. The maximum WTX value is `0x3B` [Int08], this has to be changed in the chip in order to be compliant to the standard.

6.3.3. Execution Times of Cryptographic Operations

Table 6.3 provides an overview of the execution times (period of time where the cryptographic unit is active) of the different cryptographic operations. As measuring this duration is not that accurate the values in the table are the mean values of several measurements rounded to the nearest millisecond.

$\mathbf{f_{CU}}$	**INT AUTH AES**	**INT AUTH ECDSA**	**EXT AUTH AES**
	ms	**ms**	**ms**
848 kHz	47	960	31
1.7 MHz	24	480	15
3.4 MHz	12	240	8
6.8 MHz	6	120	4

Table 6.3.: Overview of the execution times of the cryptographic operations.

6.4. Testing the Different Data Rates

As mentioned in Chapter 2 of this work the chip supports three different data rates for communication with the reader: 106, 212 and 424 kbit/s.
Concerning the ISO14443-4 standard there are four standardized data rates, 106, 212, 424 and 848 kbit/s. As not all tags support all data rates there is a field in the answer-to-select (ATS) command where the tag can tell the reader which data rates it supports. The information if different data rates for sending and receiving are supported can also be found in the ATS. In the protocol and parameter selection request (PPS), which is sent from the reader to the tag, the reader informs the tag about the used data rates for sending and receiving [Int08].
Analyzing the ATS of the CRYPTA tag shows that the chip supports the three data rates

106, 212 and 424 kbit/s as well as different data rates for sending and receiving. This is the correct behavior as it matches the specification of the chip.
For testing if the communication with different data rates works, a file-read operation with all the different data rates was performed (also different data rates for sending and receiving were used). As the read operation was successful for all the configurations the chip passed this test.

6.5. Summary

As it can be seen in this chapter, functionality tests are very important for prototype chips. Of course this tests do not cover all the functionality of the chip but the most important ones for real-life applications. Tests were performed for the different file operations, for the proper function of the cryptographic unit as well as the different data rates.
The tests in this chapter show that the implementation of the chip is correct (only the WTX value has to be changed to `0x3B` or lower to meet the standard) and meets the specification.

7. SCA Results

In this chapter we present the results of the side-channel analysis attacks on the CRYPTA tag as well as on the FPGA prototype of the chip. At the beginning the measurement setups for attacking the CRYPTA tag as well as the FPGA prototype are shown. Then the results of the attacks on the FPGA prototype in evaluation mode are presented followed by the results in standard mode. As explained in Chapter 2 the implemented countermeasures are disabled in evaluation mode and enabled in standard mode. We compare standard mode and evaluation mode to show the effectiveness of the SCA countermeasures. Next the results of the attacks on the CRYPTA chip in evaluation mode are shown. It is not possible to activate the standard mode on the chip because of a bug in the implementation. Hence it is not possible to evaluate the effectiveness of the countermeasures on the chip directly. A patched version of the chip will fix this but it is not available until now. This chapter ends with a summary of the findings.

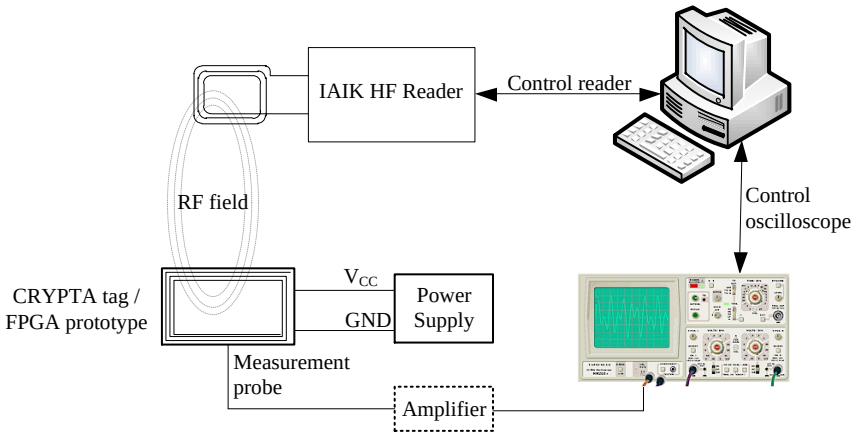

Figure 7.1.: Schematic overview of the measurement setup.

7.1. SCA Attack Setup

For the SCA attacks the oscilloscope LC584AM from LeCroy was used to record the traces. The recording process was controlled with a computer running MATLAB scripts. For the communication between the computer and the tag the IAIK HF reader was used. Figure 7.1 shows the schematic measurement setup. There were two differences in the measurement

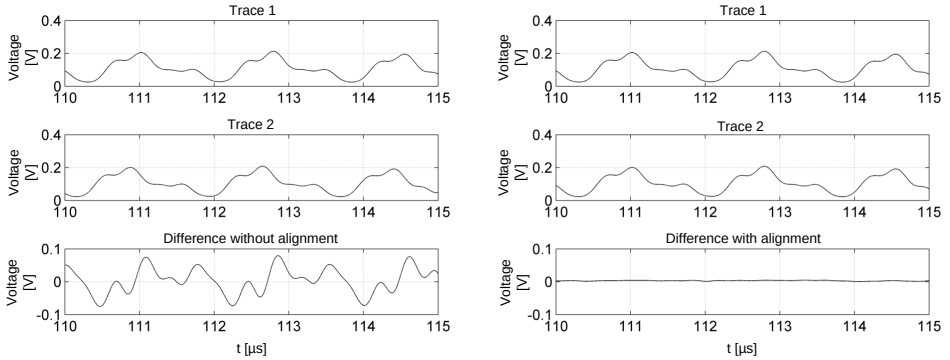

Figure 7.2.: Impact of the alignment process on the difference of two traces. The difference of the two traces on the left side was calculated before the alignment process and the difference of the traces on the right side after the alignment process. All the traces were recorded while the same data was processed.

setups for the FPGA prototype and the CRYPTA tag: the measuring method and the trigger signal.

As we did not want to modify the FPGA board to add a resistor in the power line or the ground line in order to measure the power consumption we decided to measure the electromagnetic emanation (EM) of the FPGA with an EM probe. For the measurements the following near filed probe was used: LANGER EMV-Technik, near-field probe LF B 3 [ET11]. As the amplitude of the signal from the EM probe is quite small we had to put an amplifier with a gain of 30 dB between probe and oscilloscope. A datasheet of the used amplifier can be found at [MC11]. The CRYPTA-tag prototype chip is mounted on a test board and so we could use an external power supply to power the tag. If an external power supply is connected, the chip does not extract the power supply from the reader field. So, we could use a resistor in the ground line to measure the power consumption of the chip.

As there are debug pins available on the FPGA board we used one of these pins to indicate when the cryptographic unit is running. This pin was used as a trigger for recording the EM traces. By using a dedicated pin as trigger source lead to traces that were perfectly aligned. The computational effort for the attacks decreased because the alignment step could be skipped. On the CRYPTA-tag prototype chip no debug pins are available and so the trigger signal had to be extracted from the power trace. Since the power consumption significantly increases when the cryptographic unit starts to work, extracting the trigger worked quite well. Figure 7.3 shows the power trace of a whole AES encryption including the used trigger information. However, the recorded traces were not perfectly aligned and so we had to perform an alignment step during the attack. So the computational effort for the attack was higher compared to the attack on the FPGA prototype. Figure 7.2 shows the impact of the alignment process.

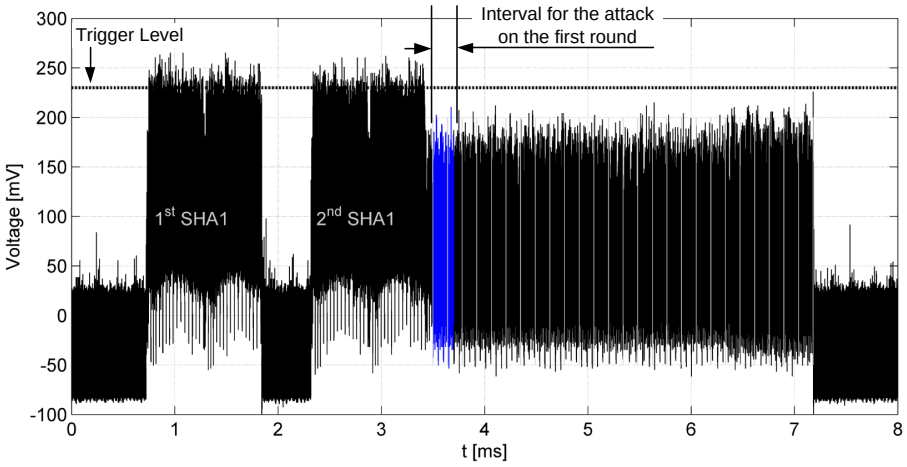

Figure 7.3.: Choice of the trigger level on the power trace of the CRYPTA tag. Also the segmentation of the different parts of the trace is shown. The highlighted part is the interesting interval for an attack on the first round of AES.

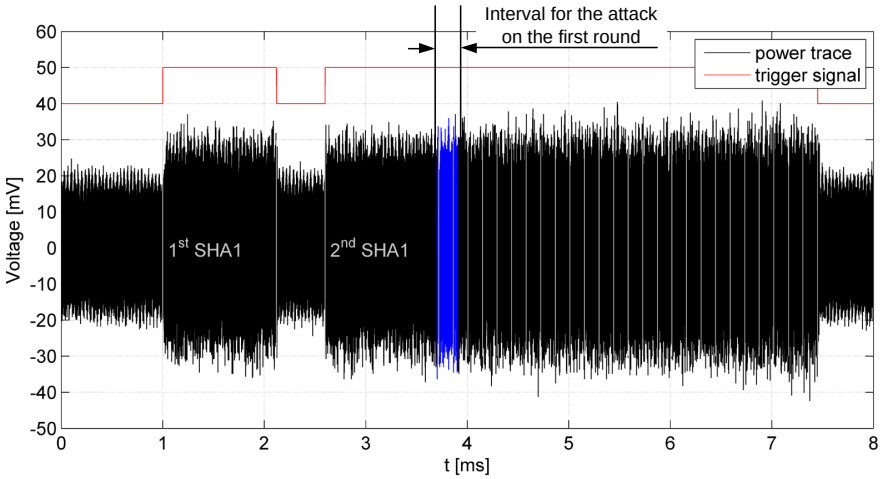

Figure 7.4.: Segmentation of the EM-trace of the FPGA prototype including the two preceding SHA1 calculations. The highlighted part of the trace is the interesting interval for attacking the first round of AES.

7.2. Attacks on the FPGA Prototype

The different steps of a DPA attack were already described in Chapter 5.4 earlier in this work. Now, we present how these steps can be used for attacking the FPGA prototype.

7.2.1. Choosing the Intermediate Result

In the first step an intermediate result has to be chosen. As we have full control over the device under attack and we know that we attack an AES implementation we could attack the result after the first round (cf. Figure 3.6) or the result before the last round (cf. Figure 3.8) of AES. The knowledge of the plaintext is a prerequisite for an attack on the first round and the knowledge of the ciphertext is a prerequisite for an attack on the last round. In order to show that both approaches lead to a correct result we performed an attack on the first round and on the last round.

7.2.2. Record the Traces

In the second step we had to record the traces. It is very important to mention that we also had to record the plaintext (sent to the tag) and the ciphertext (received from the tag) that belong to the traces in order to calculate the intermediate result in the next step.
A key issue in this step is to find the appropriate part (the time interval) in the trace where the attacked data is processed. The attack works with a higher success probability if the interval is large but the computational effort as well as the memory usage to store the traces increases. If the chosen interval is too short (the part where the attacked data is processed is not contained) the attack will fail. As we have detailed knowledge about the implemented algorithm we started with one single power trace of the whole AES encryption. This trace was then split into several parts. Before the AES encryption starts the tag performs a SHA1 calculation two times to generate a random number. The two intervals where the SHA1 calculation takes place are easy to identify in the trace. So the start of the AES encryption could be found which is directly after the end of the second SHA1 calculation. The point where the amplitude of the trace significantly decreases was considered to be the end of the AES calculation. Within these two borders the tag does the AES encryption. The attacked AES implementation computes 26 rounds in total, ten rounds of the AES algorithm itself and 16 additional rounds (dummy rounds) for the countermeasure. The whole part of the AES encryption can be split into 26 equally sized parts and for attacking the first round we put the focus on the first part. Figure 7.4 visualizes the previously mentioned points for analyzing the different parts of the power trace. The interesting part for the attack of the first round is highlighted in this trace.
For the attack on the last round of AES the tenth part of the trace had to be recorded in order to get the correct key bytes. This can be done because of the knowledge that in evaluation mode the 16 dummy rounds are inserted after the real AES calculation has finished.
For the evaluation mode a set of 200 traces was recorded for the attack on the first and the last round ($D = 200$). The length of the recording interval was $200\,\mu$s for the first attack

($4 \cdot 10^5$ samples) and was reduced down to $2\,\mu$s ($4 \cdot 10^3$ samples) in the end for a fast attack on just one key byte. A sampling rate of $2\,GS/s$ was used. The recorded traces are organized in a matrix **PT** with D rows and T columns.
For the FPGA prototype in standard mode only the first round was attacked. Our assumption was to use the same time interval as for the attack in evaluation mode with the knowledge that in average only every sixteenth trace is a valid trace, i.e. processes real data (assuming that the number of dummy rounds at the beginning follow a uniform distribution). This is because zero to 16 dummy rounds are inserted at the beginning randomly. The probability that no dummy round is inserted at the beginning and so the recorded interval contains a valid calculation is 1/16. Later in this chapter it can be seen that this interval was not the best choice for an attack in standard mode and so we had to adapt it.

7.2.3. Generating the Hypothetical Power Values

The third step and the fourth step are combined because they are strongly related to each other. First, the intermediate results have to be calculated. In order to do so the plaintexts that were previously recorded were used. Equation 7.1 shows how this intermediate result is calculated. The border for i is quite obvious as we have stored the plaintext (and the ciphertext) for every recorded trace. As the attack focuses on one key byte and the whole key space for this byte has to be covered there are 256 possible values for j. The computational effort can be reduced if the attacker knows e.g. the lowest significant bit of the attacked key byte. If this bit is zero only even values for the key byte are possible.

$$\begin{aligned} &i = 1 \ldots D \\ &j = 0 \ldots 255 \\ &\text{S-box}(plaintextByte1(i) \oplus key(j)) = intermediateResult(i,j) \end{aligned} \tag{7.1}$$

For attacking the last round of AES only the calculation of the intermediate result changes. The calculation is given in Equation 7.2 where the borders for i and j are the same as the borders in Equation 7.1. Now the ciphertext is used as input of the inverted substitution box (S-box^{-1}). The result of an attack on the last round is the tenth round key and not the AES secret key! In order to get the AES secret key the inverse key schedule has to be used. One disadvantage of this attack is that all the key bytes of the tenth round have to be found before the inverse key schedule can be used to calculate the AES key.

$$\begin{aligned} &i = 1 \ldots D \\ &j = 0 \ldots 255 \\ &\text{S-box}^{-1}(ciphertextByte1(i) \oplus key(j)) = intermediateResult(i,j) \end{aligned} \tag{7.2}$$

With this intermediate result we calculated the hypothetical power values using an appropriate power model. The used power model was the Hamming-weight model (HWM). It is a simple

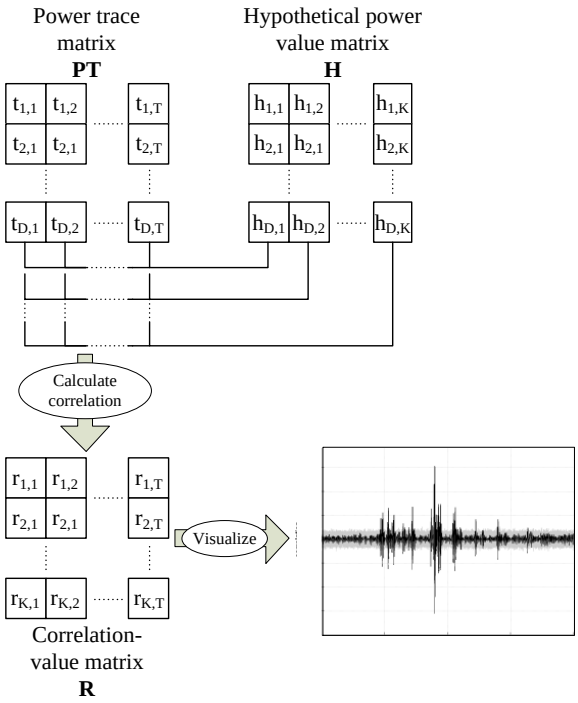

Figure 7.5.: Visualisation of the correlation steps.

model that assumes that the higher the number of logical ones in the intermediate results is the higher is the power consumption. So, the number of ones in the intermediate results has to be counted to obtain the hypothetical power value (e.g.: $0110 \rightarrow HW2$). In the next step we will see that this model fits quite well for the attack. In [SM07] also several other power models are presented. For every single intermediate result an appropriate hypothetical power value is calculated and organized in the matrix **H**. This matrix has D rows and in our case $K = 256$ columns as we use all the 256 key hypotheses (see also Figure 5.14).

7.2.4. Comparing the Hypothetical Power Values to the Recorded Traces

The last step is the computationally most expensive one. The recorded and preprocessed traces stored in the matrix **PT** are correlated with the matrix **H** containing the hypothetical power values. This step is shown in Figure 7.5. A preprocessing of the traces was necessary in order to get satisfying results.
As mentioned earlier in this chapter there was no need for an alignment process for the FPGA prototype because of the accurate trigger signal. The next preprocessing step was filtering the traces. This step could not be skipped as the test environment was quite noisy (e.g. carrier signal of the reader). The selection of the filter type (lowpass, highpass, bandpass) and

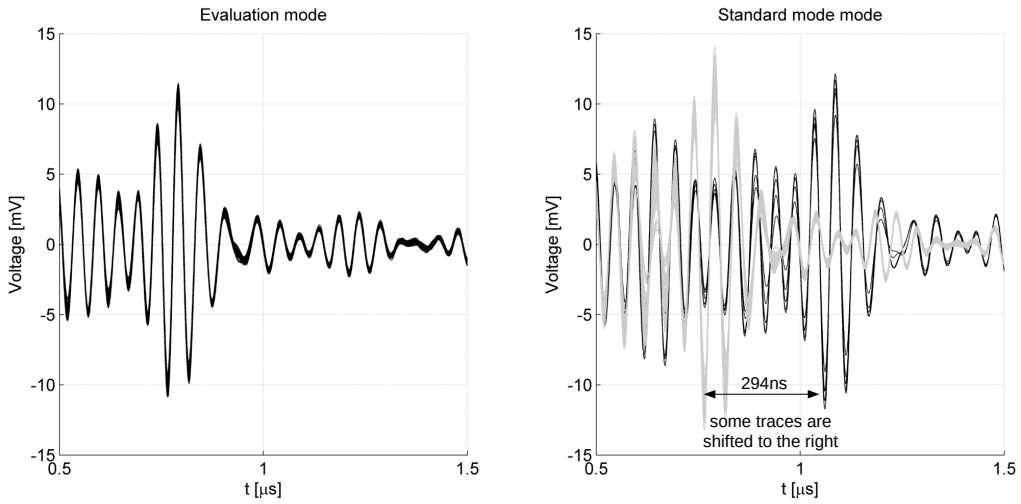

Figure 7.6.: Filtered traces recorded in evaluation mode (left) and in standard mode (right).

the filter parameters was made by inspecting the frequency spectrum of the recorded traces. It figured out that a bandpass with a start frequency of $15\,MHz$ and an end frequency of $25\,MHz$ was the best choice for the attacks on the FPGA prototype.
For the attack on the FPGA prototype in evaluation mode filtering was the only necessary preprocessing step. After that step the correlation values could be calculated. In Chapter 5.4 a description how the calculation works can be found. The correlation values were stored in a matrix $\mathbf{R}$ with $K{=}256$ rows and T columns. The correct key hypothesis was found by searching the maximum value in the matrix $\mathbf{R}$. In order to visualize the result we plotted each row of the matrix as a single graph. The row with the maximum absolute value (correct key hypothesis) was plotted in black and the other rows (wrong key hypotheses) were plotted in gray.

For the attack on the FPGA prototype in standard mode we performed some extra preprocessing steps in order to minimize the impact of the countermeasures. The aim of the first additional preprocessing step was to reduce the impact of the dummy rounds. We recorded the traces of the standard mode in the same interval as the traces in evaluation mode. By comparing the traces of the different modes we found a characteristic which seemed to give us the ability to filter out the dummy rounds. Figure 7.6 shows a small section of the 100 traces recorded in evaluation mode on the left side and the same section of 100 traces recorded in standard mode on the right side. As it can be seen in the plot of the standard mode some traces are shifted to the right. The delay is $294\,ns$ which equals two clock cycles for the used clock frequency of $6.8\,MHz$. As this shift only appears in standard mode this might be an indicator for a dummy round. We started to count the traces that were shifted to the right and the results can be found in Table 7.1.

As it can be seen in the table the ratio of the number of traces shifted to the right to the number of traces overall is close to 1/16 (=0.0625). Our first thought was that the traces shifted to the right belong to a calculation where no dummy rounds were inserted at the beginning. After several attacks without success we started to shift the recording interval to

Number of overall traces	Number of traces shifted to the right	Ratio
100	5	0.0500
1000	65	0.0650
10000	622	0.0622

Table 7.1.: Number of traces recorded overall in comparison to the number of shifted traces.

the right. It figured out that the shift of the traces indicates that 16 dummy rounds (=the maximum number of dummy rounds) were inserted at the beginning. The ability to detect dummy rounds decreases the attack complexity a lot. The only active countermeasure after sorting out the traces with dummy rounds is shuffling. In the case of shuffling the probability that the first byte is processed at the first position is again 1/16 as the AES state has 16 bytes and every byte can be processed at any position. The correlation coefficient in this case would decrease by a factor of 16 and the number of needed traces would increase by about 16^2=256 compared to an attack without countermeasures [PHF].
In order to decrease the impact of shuffling we introduced another preprocessing step: windowing. As it can be read later in this chapter this preprocessing step requires knowledge of the exact points in time where specific operations take place. In the best case the correlation coefficient increases by a factor of $\sqrt{16}$ and the number of needed traces decreases by a factor of 16.

7.2.5. Results for Attacks on the First Round in Evaluation Mode

For the attacks in evaluation mode we started with a large recording interval of 200 μs to be sure to cover the whole first round. Figure 7.7 shows the result of the attack on the first key byte. The correct value could be found with less than 100 traces and the correlation coefficient of 0.6 shows that the used power model fits well for the FPGA prototype.
The long recording interval leads to a quite long computation time of the attack and so the length of the interval was decreased to 20 μs. The computation time could be reduced by a factor of ten. In further consequence we reduced the recording interval to 2 μs which led to a speedup of ten again. A prerequisite for a successful attack on a short interval like 2 μs is the knowledge of the exact position where the attacked intermediate result is processed.
Table 7.3 gives values for the time where the maximum correlation for every byte occurs ($t_{maxcorr}$) as well as the value of the maximum correlation ($maxcorr$). The table is sorted in ascending order of $t_{maxcorr}$. By analyzing the values in the table several interesting observations can be made:

- The processing order of the bytes in the state is diagonal. The reason for this processing order is that the shift-rows round transformation does not have to be implemented. The bytes of the state have to be stored in the correct position at the end of the round.
- The first key byte has the best leakage for the DPA attack as the correlation value is nearly twice as big (0.611) compared to the values of the other key bytes (0.3 in average). An analysis of the reason for this behaviour would be out of the focus of this

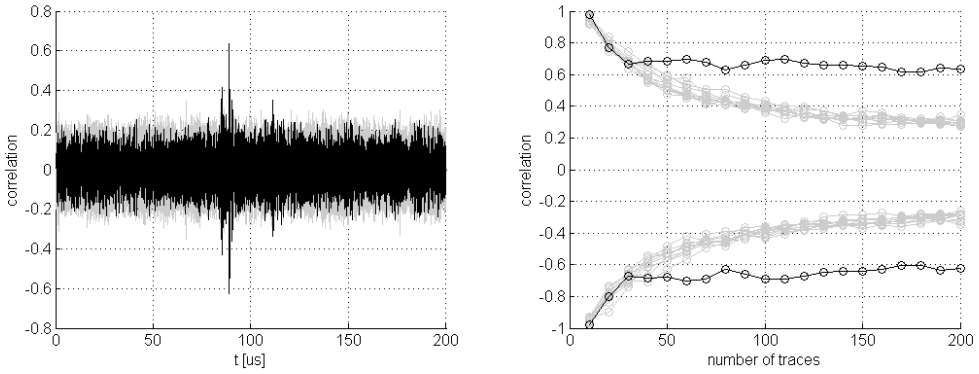

Figure 7.7.: DPA attack on the first key byte of the FPGA prototype tag (left) and the correlation coefficient as a function of the number of traces (right).

work and might be addressed in future. Considering this observation the best choice for the attacks in standard mode will be the first key byte.

The correlation traces for every single attacked key byte can be found in Appendix A (Figure A.1 and Figure A.2).

7.2.6. Results for Attacks on the Last Round in Evaluation Mode

In this section we want to put the focus on the results of the attack on the last round of the implemented AES algorithm. Finding the correct interval for this attack was more difficult: Earlier in this chapter the assumption was that the last round is processed in the 10^{th} segment of the trace in Figure 7.4 but the attack on this interval did not produce any correct result. So the recording interval was shifted to the right and for the 15^{th} segment we got the correct values for the 10^{th} round key. A detailed analysis of the AES implementation would be necessary to find the reason for this deviation compared with our assumption. The SCA leakage for the attack on the last round appears about 2.2 ms after the AES encryption has started.
Figure 7.8 shows the result of the attack on the first key byte of the tenth round. The results for the other key bytes look similar. With less than 200 traces all correct key bytes can be found.
Compared to the attack on the first round the average correlation value is higher.
Table 7.3 gives values for the time where the maximum correlation occurs ($t_{maxcorr}$) as well as the value of the maximum correlation ($maxcorr$) for every key byte. In the table it can be seen that the order of the appearance of the leakage of the key bytes for the attack on the last round is different compared to the order for the attack on the first round. The order of execution is different here because also the attack direction is reversed (from the ciphertext to the intermediate result). So also the direction of shift rows is inverted.

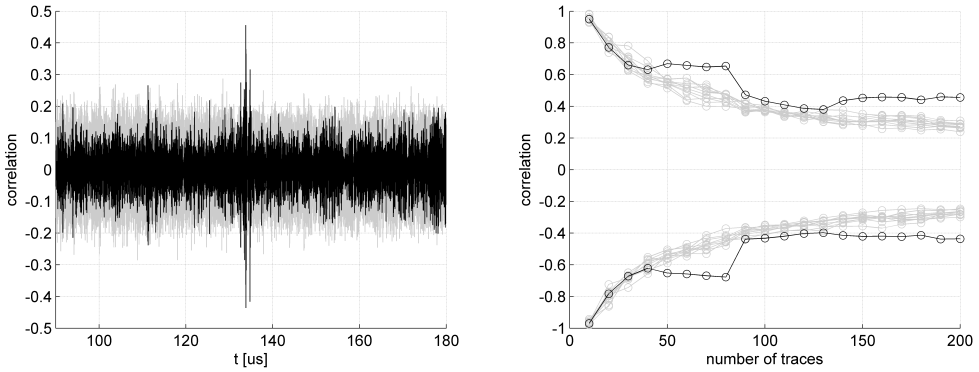

Figure 7.8.: DPA attack on the first key byte of the tenth round key (left) and the correlation coefficient as a function of the number of traces (right).

7.2.7. Results for Attacks in Standard Mode

Before we started with the attacks on the FPGA prototype in standard mode we did some estimations based on the result for the attack on the first key byte in evaluation mode. These estimations as well as the results of the equivalent attacks can be found in Table 7.2. The reason for this step was to see how much effort is needed for a successful attack and how much traces have to be recorded.

	Correlation value	**Estimated number of traces**	**Number of traces for real attack**
No active countermeasures	0.61100	75	75
Only shuffling (windowing used)	0.12775	1 200	5 000
Only shuffling (no windowing)	0.03194	19 200	10 000
Shuffling and dummy rounds*	0.03194	307 200	160 000
Shuffling and dummy rounds	0.00199	$> 4\,900\,000$	not performed

Table 7.2.: Estimate of the required number of traces for a DPA attack when different countermeasures are activated (*. . . Dummy round detection active).

Without countermeasures (evaluation mode) a DPA attack using 75 traces lead to a correct result. According to Equation 5.7, the noise floor when using 75 traces is 0.462 and the gap to 0.611 is big enough to filter out the trace belonging to the correct hypothesis.
In standard mode both countermeasures, the dummy rounds and shuffling are activated. The correlation value indicating the correct key value decreases by a factor of 256 in that case. Subsequently the number of traces increases by a factor of 256^2. This leads to an estimated number of nearly 5 million traces required for a successful attack. As our average record speed for the attacks was one trace per second, this would lead to a measurement time of nearly 60 days. As mentioned earlier in this chapter a pattern was found to filter out the dummy rounds. So, after filtering out the dummy-round traces the size of the reduced set

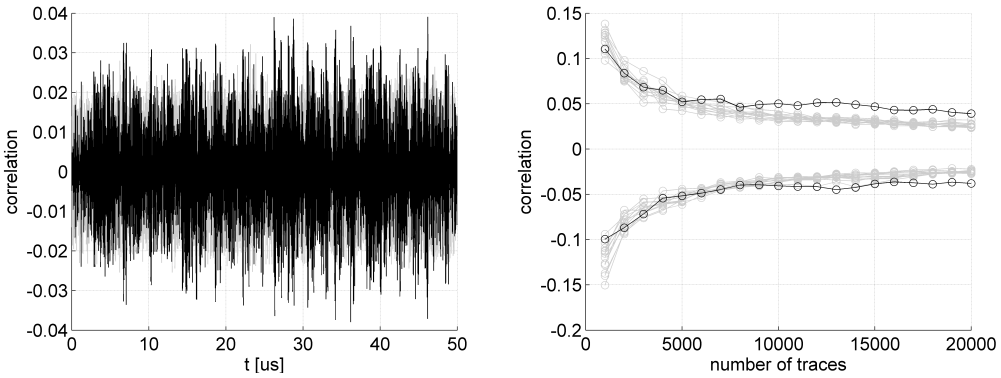

Figure 7.9.: DPA attack in standard mode on the first key byte (left) and the correlation coefficient as a function of the number of traces (right).

of recorded traces decreases by a factor of 16 compared to the basic set but it is ensured that the expected data is processed in this interval. For this smaller set of traces it holds that shuffling is the only countermeasure. The reduced set of power traces has to contain at least 19 200 traces to perform a successful attack. With that knowledge we could calculate the number of required traces in the basic set, which is $16 \cdot 19\,200 = 307\,200$ traces. With a recording speed of one trace per second this leads to a total measuring time of 85 hours. Compared to an attack without dummy-round detection this is 16 times faster.
Figure 7.9 shows the result of the attack on the first key byte in standard mode. The basic set contained 320 000 traces and after filtering out the dummy rounds we got a bit more than 20 000 traces. With this reduced set of traces we did a DPA attack and were able to reveal the correct key byte. The maximum correlation value of the correct key trace is about 0.039 and is even a bit higher compared to the estimated one. Also the number of traces needed for the successful attack is close to the estimated value. Compared to the attacks without countermeasures there is no single peak in the key trace corresponding to the correct key byte. This is because shuffling spreads the single peak on 16 different points in time. The more traces are used the better can these 16 points be identified.

In a last experiment we tried to use windowing as an additional preprocessing step to reduce the impact of shuffling. Before we did this preprocessing step the traces with the dummy rounds were filtered out. As a result, less traces should be required for a successful DPA attack. Windowing means to add up the values around the 16 points in the power traces where the attacked intermediate result can appear. A detailed description of windowing can be found in the book of Mangard et. al [SM07]. It figured out that the most difficult step here was to find an appropriate window pattern. The window pattern denotes the indices of the points in the trace that are added up. In order to find such a pattern we took the correct key trace from the successful attack above and picked out the indices of the highest peaks in this trace. The result of this attack can be found in Figure 7.10. In this figure we have also plot the evolution of the correlation coefficient of the attack without windowing for comparison. It can be seen that the windowing step does not increase the correlation coefficient, it is even smaller for a small number of traces. This degradation occurs because

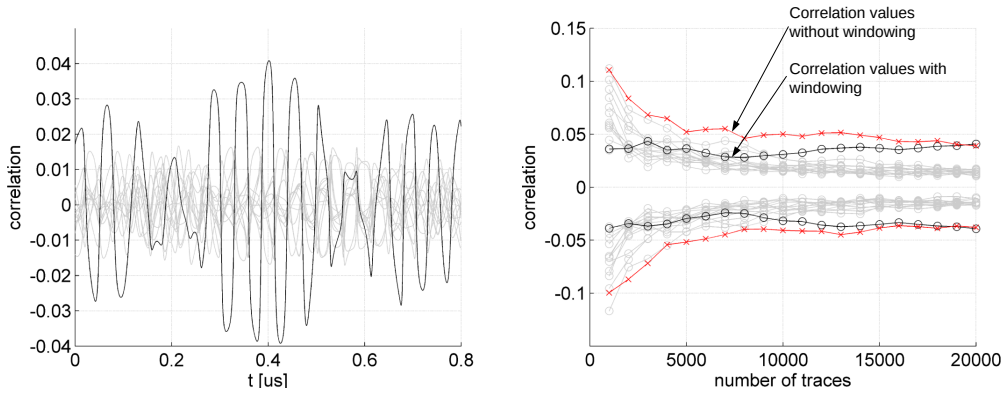

Figure 7.10.: DPA attack (with windowing) in standard mode on the first key byte and evolution of correlation coefficient with increasing number of traces used.

of a bad choice of the used window pattern.

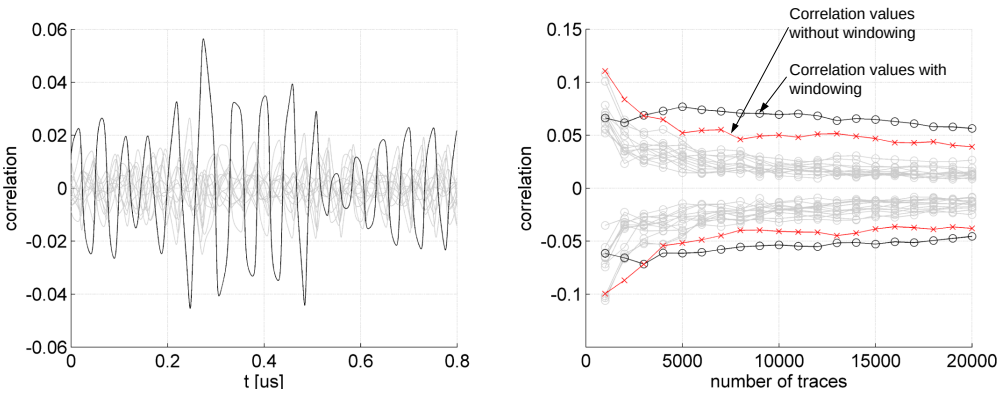

Figure 7.11.: DPA attack (with windowing) in standard mode on the first key byte and evolution of correlation coefficient with increasing number of traces used.

As the upper result was not satisfying we performed another attack with a different window pattern. The result can be found in Figure 7.11. Here it can be seen that the usage of windowing improves the attack. Only 5 000 traces are needed in order to reveal the correct key byte. Compared to the attack without windowing this is a big improvement. Our estimations (1 200 traces for a successful attack) from Table 7.2 could not be reached. The reasons for the deviation is that it is hard to find a suitable pattern without an exact knowledge and analysis of the timing of the chip.

7.3. Attacks on the CRYPTA Tag

In this section the detailed steps of the attack on the CRYPTA tag are explained. First the five steps of the attack strategy are described followed by the results of the attack on the

first round and the last round in evaluation mode.

7.3.1. Choosinging the Intermediate Result

On the CRYPTA tag the same algorithm is attacked as on the FPGA prototype. Because of that fact the same assumptions made in Section 7.2.1 can also be used for the first step here.

7.3.2. Recording the Traces

In the second step we had to record the power traces as well as the plaintexts sent to the tag and the ciphertext received from the tag. We started to record a single trace of the whole AES encryption to split it up in the different parts. This gave us an overview of which parts of the trace might be interesting for the attack.
We were able to identify several parts by inspecting this single trace: At the beginning there is the SHA1 calculation two times to generate a random number on the tag. After the second SHA1 calculation the AES encryption starts. The end of the AES encryption (when the cryptographic unit stops working) can be easily identified by the drop of the amplitude of the trace. As the starting point and the end point of the AES encryption were found this interval could be split into 26 equally sized parts (10 real AES rounds and 16 dummy rounds). For attacking the first round of AES we only had to focus on the first of these 26 intervals as in evaluation mode no dummy rounds are located before the first real round. Figure 7.3 graphically shows how the trace is splitted into the above mentioned parts.
For the evaluation mode a set of 200 traces was recorded ($D = 200$). A recording interval of $5\,ms$ ($T = 5 \cdot 10^5$ samples $\Rightarrow 100\,\mathrm{kS/s}$ sampling rate) was used for the first attack to cover the whole algorithm. So the correctness of the trace splitting mentioned above could be verified. For further attacks the recording interval was then reduced to smaller values to minimize the computational effort and the memory requirements. As a result we could also increase the sampling rate. The best results could be achieved with a sampling rate of $2\,\mathrm{GS/s}$, which was used for all the attacks with a shorter recording interval.
For the CRYPTA tag in evaluation mode also attacks of the last round of AES were performed. The set of power traces with the recording length of $5\,ms$ was used for this attack.

7.3.3. Generateing the Hypothetical Power Values

A detailed explanation how to generate the hypothetical power values can be found in Section 7.2.3. As the attacked algorithm is the same for the FPGA prototype and the CRYPTA tag the calculation of the intermediate result is also the same.
After the first attack it figured out that also the Hamming-weight model as power model worked well for attacking the CRYPTA tag.

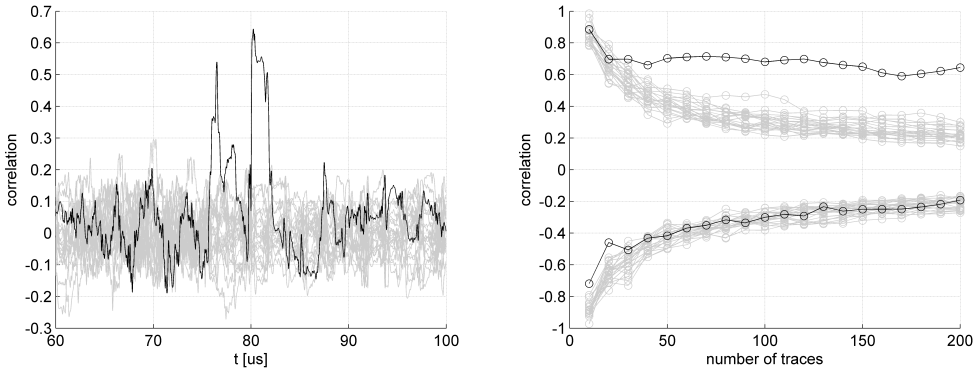

Figure 7.12.: DPA attack on the first key byte of the CRYPTA tag (left) and the correlation coefficient as a function of the number of traces (right).

7.3.4. Comparing the Hypothetical Power Values to the Recorded Traces

Figure 7.5 shows a graphical visualization of this step. In order to get satisfying results the recorded traces had to be preprocessed before the correlation matrix $\mathbf{R}$ could be calculated. As a first preprocessing step the traces had to be filtered in order to get rid of unwanted parts in the trace like the reader signal and surrounding noise. After analyzing the frequency spectrum of the power traces it figured out that a lowpass filter with a cut-off frequency of 6 MHz was a good choice.
As mentioned in Section 7.1 no trigger pin was available on the CRYPTA tag. Because of that we had to extract the trigger information from the power trace. Compared to a dedicated trigger pin this method was not that accurate. The result was that the traces were not good enough aligned to get correct results. So we had to align the traces in a second preprocessing step. After the SHA1 calculation and before the first round of AES the chip performs some operations which are equal every time. This part of the traces could be used as the alignment pattern for this step.
For the CRYPTA tag in evaluation mode the two preprocessing steps mentioned above were the only steps which had to be performed for a successful attack. With the preprocessed traces the correlation matrix $\mathbf{R}$ could be calculated.
The standard mode on the CRYPTA tag could not be activated and as a result no further preprocessing steps (e.g. detecting dummy rounds, windowing) for decreasing the impact of the countermeasures are presented here.

7.3.5. Results for Attacks on the First Round in Evaluation Mode

Figure 7.12 shows the result of the attack on the first key byte. The high correlation value of 0.65 illustrates that the used power model (Hamming weight) fits well to the attacked device. In the right plot it can be seen that the correct key byte can be found with less than 100 traces.

Attacks were performed on all of the 16 key bytes and for every byte the correct value could

be found with the DPA attack with less than 200 traces. Figure A.3 and Figure A.4 show the correlation traces for every single key byte. Every plot has one key trace with a significant peak indicating the trace corresponding to the correct key hypothesis. Table 7.4 gives an overview of the points in time where the maximum correlation values appear in the traces. The table is sorted in ascending order of $t_{maxcorr}$.

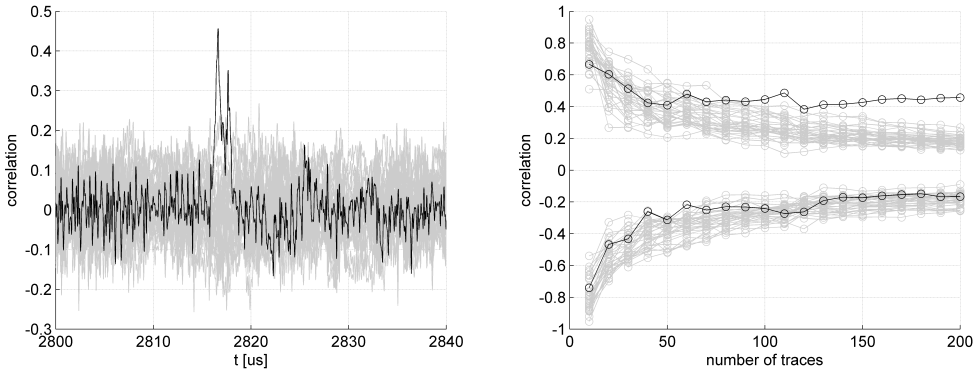

Figure 7.13.: DPA attack on the first key byte of the tenth round key (left) and the correlation coefficient as a function of the number of traces.

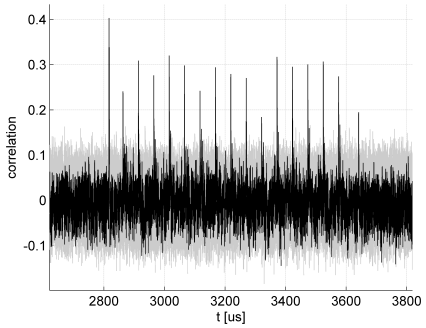

Figure 7.14.: Correlation peaks in the trace of the attack on the first key byte of the tenth round key of the CRYPTA chip.

7.3.6. Results for Attacks on the Last Round in Evaluation Mode

Figure 7.13 shows the result of the attack on the first key byte of the tenth round key. As the correlation value is about 0.4 the correct key value could be found with less than 200 traces. With this attack all key bytes of the tenth round key could be revealed.
Analyzing the correlation trace of the correct key hypothesis, several significant and equidistant peaks can be identified. Figure 7.14 visualizes this discovery. For the other attacked key bytes the same number of peaks occurs in the correlation trace corresponding to the correct hypothesis. The first peak is always the highest one except for key byte 11. In the case of key byte 11 the last peak is the highest one. Detailed knowledge about the implementation

of the chip would be required in order to find the reason for the peaks.
In Table 7.4 the points in time where the highest correlation occurs $t_{maxcorr}$ as well as the maximal correlation value *maxcorr* for every key byte are listed.

7.4. Summary of the Results

As it was shown in this chapter the implementation of a secure algorithm like AES on a hardware device does not imply that it is hard to find the secret key. If no countermeasures against DPA attacks are implemented an attacker can reveal the key in a few minutes as shown above. The better the knowledge about the implemented algorithm is, the faster the attack can be performed. We were able to record about 60 traces per minute. Without countermeasures the required amount of traces for a successful attack could be recorded in about three minutes. With this set of traces it is then possible to reveal the whole secret key.
We also showed in this chapter that the used countermeasures against DPA attacks implemented on the CRYPTA tag (the FPGA prototype respectively) increase the attack complexity a lot. It is much more difficult to find the secret key when the countermeasures are activated. The correct key in standard mode could only be found with high effort. If the attacker is able to filter out the dummy rounds and also a good window pattern is known it is possible to find the key bytes of the AES key with a set of $5\,000 \cdot 16 = 80\,000$ traces. Compared to the results in evaluation mode, where the secret key could be revealed using 75 traces, the attack complexity increases by a factor of 1 066 (80 000/75).
If the insertion of dummy rounds would be the only implemented countermeasure, this would increase the attack complexity only by a factor of 16 because of a pattern in the traces which can be used to filter out the dummy rounds.
If shuffling would be the only implemented countermeasure about 5 000 traces are needed to reveal the secret key in case that a good window pattern is known (cf. Figure 7.11). So the attack complexity would increase by a factor of 67 (5 000/75) compared to the evaluation mode.
If no methods would work to decrease the impact of the countermeasures about 5 million traces would be needed in order to perform a successful attack. This means that the attack complexity would increase by a factor of 66 000 (5 000 000/75).
In this chapter we also stated the importance of the correct interval for recording traces to perform a DPA attack. For the attack on the last round of AES on the FPGA prototype the wrong recording interval was used at the beginning because of incorrect assumptions. As a result the key bytes could not be revealed. Several different recording intervals had to be used until we got the results presented in Section 7.2.6 of this chapter.

	First round		**Last round**	
Key Byte	$\mathbf{t_{maxcorr}}$	**maxcorr**	$\mathbf{t_{maxcorr}}$	**maxcorr**
	μ**s**		μ**s**	
1	0	0.611	0	0.456
6	2.238	0.323	25.760	0.422
11	4.008	0.260	3.566	0.317
16	5.781	0.214	30.807	0.367
5	12.279	0.322	11.730	0.520
10	14.048	0.317	38.687	0.348
15	15.820	0.284	15.553	0.387
4	17.721	0.285	40.964	0.393
9	24.044	0.297	23.917	0.453
14	26.199	0.243	1.828	0.325
3	27.903	0.326	27.524	0.387
8	29.670	0.311	5.535	0.387
13	35.991	0.313	35.865	0.465
2	37.760	0.241	13.375	0.440
7	39.531	0.278	39.161	0.470
12	46.345	0.287	17.347	0.428
average:		0.301		0.410

Table 7.3.: Occurrence of the leakage of the single key bytes and the correlation values for the FPGA prototype.

	First round		**Last round**	
Key Byte	$\mathbf{t_{maxcorr}}$	**maxcorr**	$\mathbf{t_{maxcorr}}$	**maxcorr**
	μ**s**		μ**s**	
1	0.000	0.490	0.000	0.406
6	1.776	0.340	25.570	0.384
11	3.656	0.240	833.660	0.457
16	5.330	0.270	29.050	0.512
5	11.952	0.297	12.940	0.453
10	13.718	0.357	37.630	0.481
15	16.526	0.215	16.480	0.398
4	17.389	0.233	41.090	0.473
9	23.906	0.400	23.890	0.507
14	25.670	0.329	1.550	0.562
3	27.446	0.347	28.430	0.514
8	29.225	0.309	5.420	0.431
13	35.853	0.405	36.870	0.424
2	37.626	0.337	13.580	0.473
7	39.393	0.330	40.380	0.454
12	41.170	0.331	17.070	0.512
average:		0.327		0.465

Table 7.4.: Occurrence of the leakage of the single key bytes and the correlation values for the CRYPTA tag.

8. Conclusion

In this thesis we showed that the the CRYPTA tag works according to the specifications and that the implemented countermeasures that protect the AES implementation against SCA attacks are effective.
Three problems were encountered on the chip during the tests of the CRYPTA tag: one waiting-time extension (WTX) value is incorrect, the power supply does not work proper for the highest clock frequency of the chip and it is not possible to activate the SCA countermeasures on the chip. Solutions for all the problems were found during the work.
The WTX value sent from the tag to the reader to request a longer timeout for the ECDSA calculation is not according to the standard. The value used by the tag is `0x40` but the highest possible value that is allowed is `0x3B`. We corrected the error on the FPGA implementation of the tag and proved that after this modification the ECDSA signature generation could be performed without any problems. All the file-protection mechanisms to prevent unauthorized access work according to the specification.
During the tests of the cryptographic unit we found the problem concerning the power supply of the chip. When the chip runs with the highest clock frequency (6.8 MHz) the power consumption is too high to finish the cryptographic operations. We solved the problem by inserting a small capacitor between power and ground line of the chip. This problem does not directly relate to the functionality of the chip but to the power supply of the chip.
During the security evaluation of the AES implementation the last problem concerning the activation of the SCA countermeasures was found. The tag can work in evaluation mode and in standard mode. The difference is that in evaluation mode the countermeasures are disabled and in standard mode they are enabled. First we performed SCA attacks in evaluation mode where the correct key bytes could be found with less than 200 traces. When we switched to standard mode we got the correct results with the same number of traces. However, the attack complexity in standard mode should be much higher. The reason for that behaviour was that the parameter byte for countermeasures was not a random value but `0x00` every time. So it was not possible during this work to test the CRYPTA tag with activated countermeasures. We corrected the bug in the FPGA implementation and could use the FPGA prototype in order to evaluate the effectiveness of the implemented countermeasures.
We performed an SCA attack on the FPGA prototype in evaluation mode and compared the result with the results we had for the CRYPTA tag. Again, less than 200 traces were needed to reveal the secret key. Also the timing was the same so it can be said that both devices work equal and the results are comparable. Next, we switched to the standard mode on the FPGA prototype. We performed the same attack as in evaluation mode but we could not reveal any information about the key with 200 traces. We did an estimation with the result that about five million traces are needed to perform a successful attack with activated countermeasures. As this would lead to an attack time of 60 days we decided to search for ways to make attacks

against countermeasures more effective. An advantage in this step was that we had detailed knowledge about the chip. After analyzing the recorded traces in standard mode we found a pattern to detect dummy rounds. This gave us the ability to decrease the impact of the dummy-round countermeasure. With this knowledge we did another estimation and the result was that 307 200 traces are necessary for a successful attack. We recorded a set of traces with the necessary size and as a result we could reveal the secret key. The time for recording the traces reduced from 60 days to 85 hours. After another improvement we were able to perform a successful attack with 80 000 traces. Here, we additionally used windowing to decrease the impact of the shuffling countermeasure.
The results of the SCA attacks show on the one hand that with some preprocessing steps the impact of the countermeasures can be decreased. On the other hand the attack complexity compared to the attacks without countermeasures still increases a lot. It must also be mentioned that we had detailed knowledge about the chip which was required for the preprocessing steps. An attacker normally does not have such a detailed knowledge about the attacked device and so the effort for a successful attack increases.
Future work should focus on the implementation of the dummy rounds. If it is not possible to detect the dummy rounds the attack complexity can be increased a lot. Also an evaluation of the real CRYPTA tag should be done when a patched version of the chip is available. It should be verified that the behaviour of the real chip in standard mode equals the behaviour of the FPGA prototype. Also the analog frontend (AFE) of the tag should be re-engineered in order to guarantee the power supply of the chip for the highest clock frequency (6.8 MHz).

A. DPA plots

In this appendix the correlation traces for the attacks on all the 16 key-bytes can be found. The results are from attacks on the first round using the FPGA prototype as well as the CRYPTA chip.
Figure A.1 shows the results of the DPA attack on the FPGA prototype for the key-bytes one to eight and Figure A.2 shows the results of the DPA attack on the FPGA prototype for the key-bytes nine to sixteen. The trace corresponding to the correct key hypothesis is plotted in black and the wrong hypotheses are plotted in gray.
Figure A.3 shows the results of the DPA attack on the CRYPTA chip for the key-bytes one to eight and Figure A.4 shows the results of the DPA attack on the CRYPTA chip for the key-bytes nine to sixteen. The trace corresponding to the correct key hypothesis is plotted in black and the wrong hypotheses are plotted in gray.

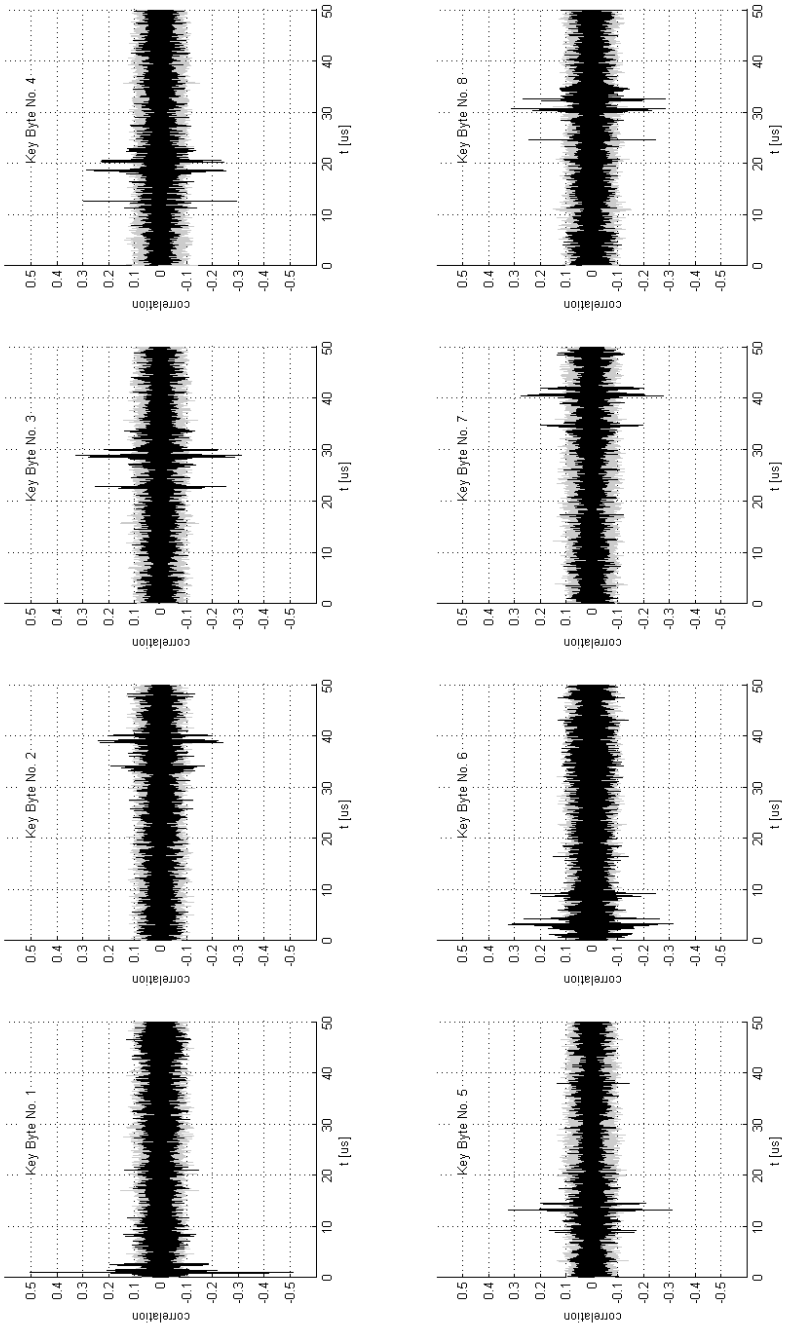

Figure A.1.: Correlation traces of key byte 1 to key byte 8 (FPGA prototype).

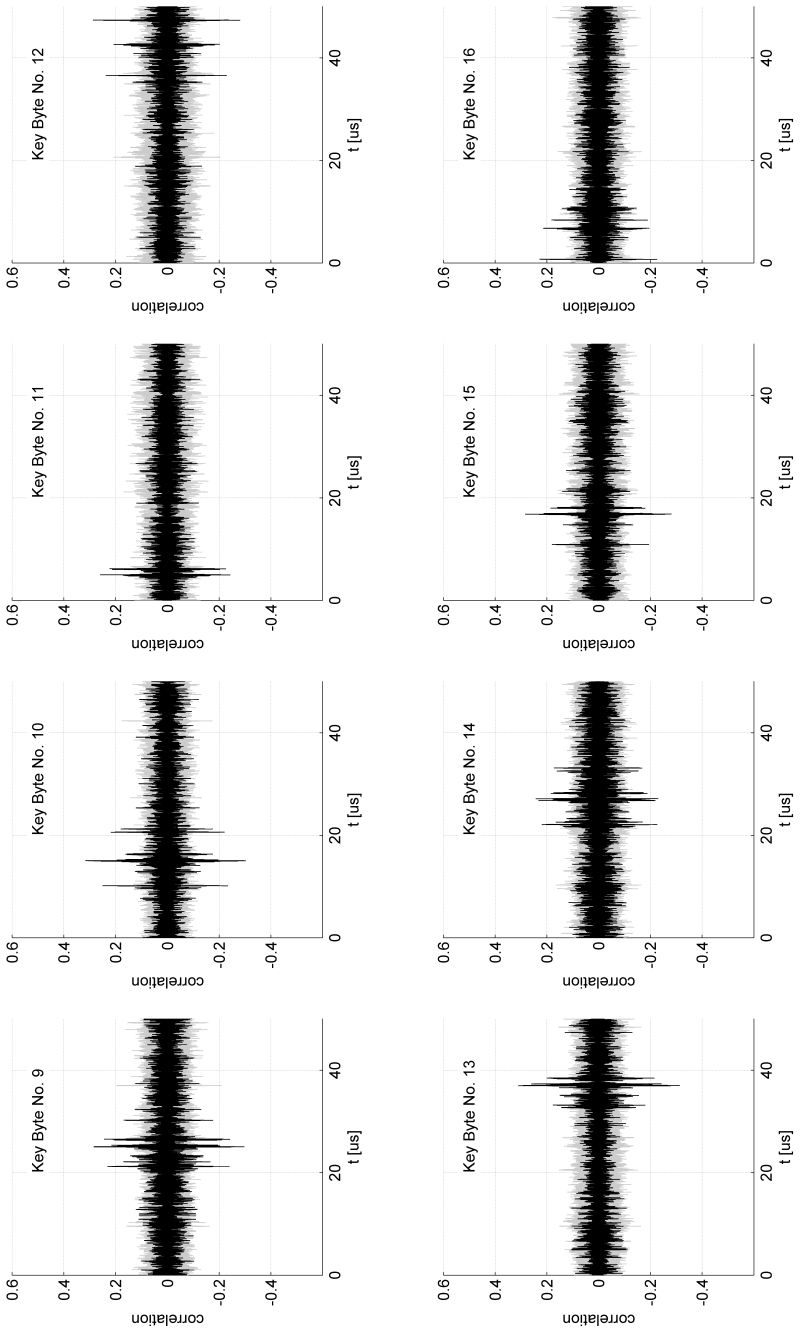

Figure A.2.: Correlation traces of key byte 9 to key byte 16 (FPGA prototype).

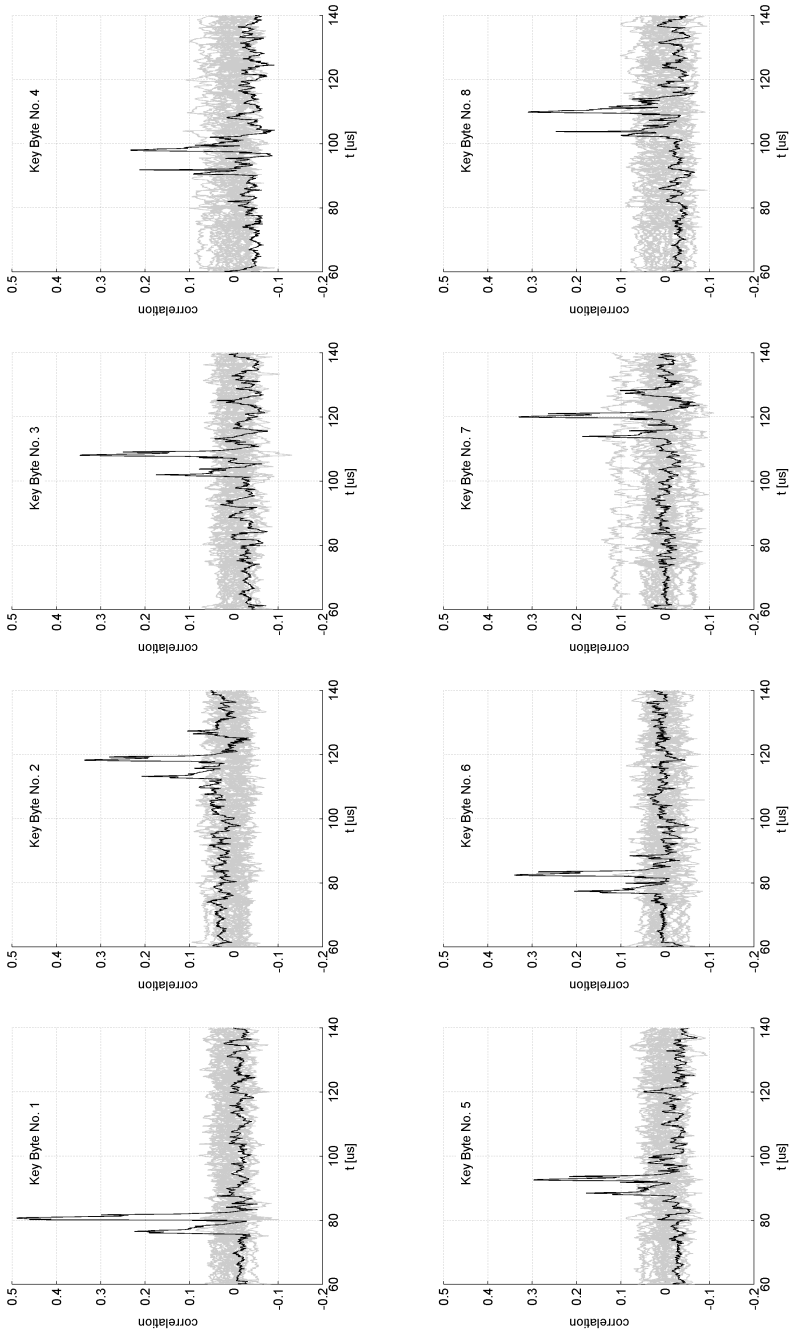

Figure A.3.: Correlation traces of key byte 1 to key byte 8 (CRYPTA tag).

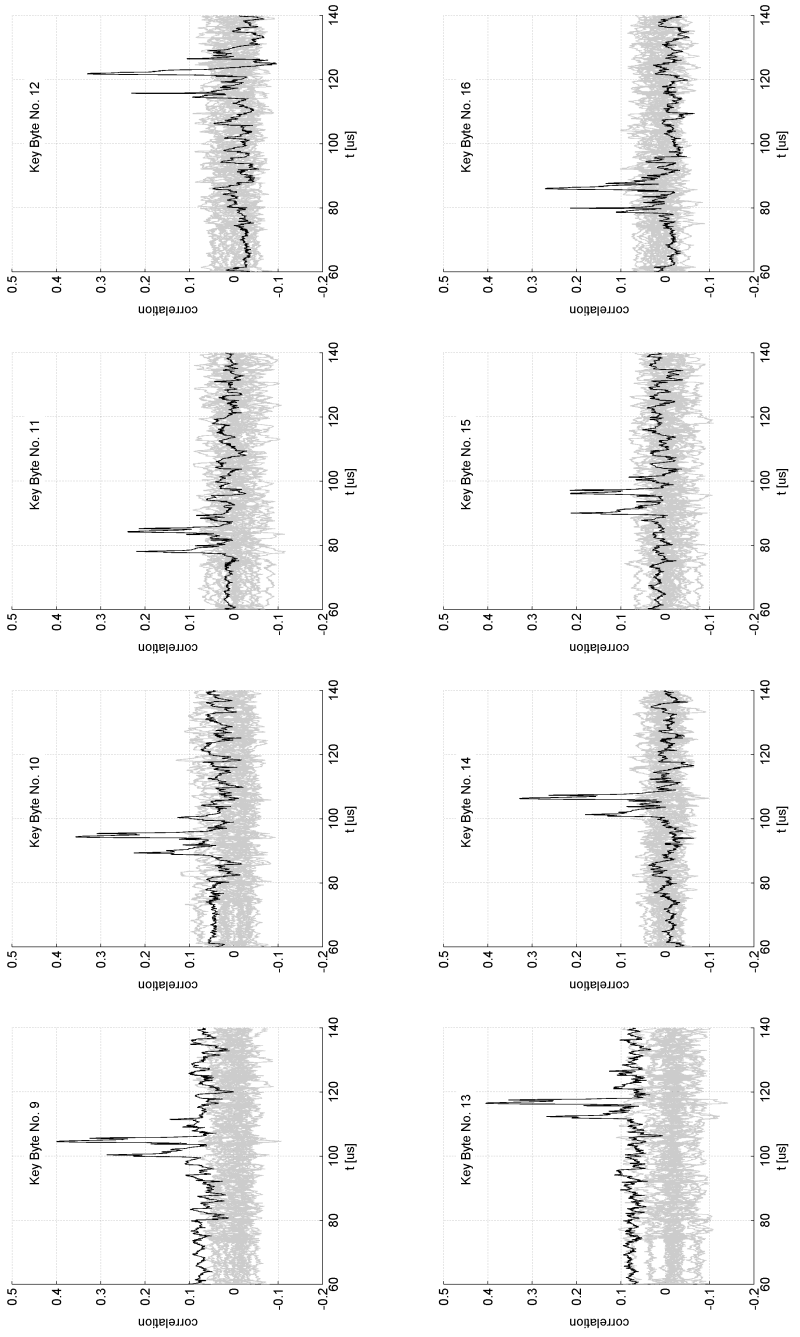

Figure A.4.: Correlation traces of key byte 9 to key byte 16 (CRYPTA tag).

List of Abbreviations

AES	Advanced Encryption Standard
AFE	Analog Frontend
APDU	Application Protcol Data Unit
ATS	Answer To Select
CC	Capability Container
CFG	Configuration
CRYPTA	Cryptographic Protected Tag
DES	Digital Encryption Standard
DLP	Discrete Logarithm Problem
DPA	Differential Power Analysis
DSA	Digital Signature Algorithm
EC	Elliptic Curve
ECC	Elliptoc Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDSA	Elliptic Curve Digital Encryption Standard
EEPROM	Electrical Eraseable Programmeable Read Only Memory
EM	Electromagnetic Emanation
FIPS	Federal Information Processing Standard
FPGA	Fiels-Programmeable Gate Array
FWT	Frame Waiting Time
GND	Ground
HD	Hamming Distance
HF	High Frequency
HLTA	Halt A
HW	Hamming Weight

HWM	Hamming Weight Model
IAIK	Institute for Applied Information Processing and Communications
ISO	International Organisation for Standardization
NDEF	NFC Data Exchange Format
NFC	Near-Field Communication
NIST	National Institute of Standards and Technology
PPS	Protocol Parameter Selection
RACK	Receive Ackonwledge
RCU	RFID Control Unit
RFID	Radio-Frequency Identification
RFU	Reserved for Future Use
RNAK	Receive Not Acknowledge
RSA	Rivest Shamir Adleman
SCA	Side-Channel Analysis
SHA	Secure Hash Algorithm
SPA	Simple Power Analysis
SW	Status Word
USB	Universal Serial Bus
WTX	Waiting-Time Extension

Bibliography

[ABHW04] Harald Aigner, Holger Bock, Markus Hütter, and Johannes Wolkerstorfer. A Low-Cost ECC Coprocessor for Smartcards. In Marc Joye and Jean-Jacques Quisquater, editors, *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 107–118. Springer, 2004.

[Aus11a] A-SIT Secure Information Technology Centre Austria. Citizen card homepage. Online `http://www.buergerkarte.at`, 2011.

[aus11b] austriamicrosystems. austriamicrosystems homepage. Online `http://www.austriamicrosystems.com/`, 2011.

[BK03] Régis Bevan and Erik Knudsen. Ways to Enhance Differential Power Analysis. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2003.

[BS96] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. (September):1–14, 1996.

[CHM06] Elisabeth Oswald Christoph Herbst and Stefan Mangard. An AES Smart Card Implementation resistant to Power Analysis Attacks. pages 1–14, 2006.

[Dwo01] Morris Dworkin. Recommendation for Block Cipher Modes of Operation. *Nist Special Publication*, 2001.

[ET11] LANGER EMV-Technik. Langer emv-technik. Online `http://www.langer-emv.de/produkte/stoeraussendung/nahfeldsonden/lf-1/bauformen-wirkprinzip/`, 2011.

[FAH+10] Martin Feldhofer, Manfred Josef Aigner, Michael Hutter, Thomas Plos, Erich Wenger, and Thomas Baier. Semi-Passive RFID Development Platform for Implementing and Attacking Security Tags. In *Workshop on RFID / USN Security and Cryptography - RISC 2010, November 9-10, London, UK, 2010.*, 2010.

[FDW04] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong Authentication for RFID Systems using the AES Algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August*

11-13, 2004, Proceedings, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer, August 2004.

[Fel10] Martin Feldhofer. Crypta device specification. Revision 1.1, 2010.

[FGKV07] Wieland Fischer, Berndt M. Gammel, Oliver Kniffler, and Joachim Velten. Differential Power Analysis of Stream Ciphers. In Masayuki Abe, editor, *Topics in Cryptology - CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007, Proceedings*, volume 4377 of *Lecture Notes in Computer Science*, pages 257–270. Springer, February 2007.

[Fin03] Klaus Finkenzeller. *RFID-Handbook*. Carl Hanser Verlag, 2nd edition, April 2003. ISBN 0-470-84402-7.

[FW07] Franz Fürbass and Johannes Wolkerstorfer. ECC Processor with Low Die Size for RFID Applications. In *Proceedings of 2007 IEEE International Symposium on Circuits and Systems*. IEEE, IEEE, May 2007.

[HFP10] Michael Hutter, Martin Feldhofer, and Thomas Plos. An ECDSA Processor for RFID Authentication. In Siddika Berna Ors Yalcin, editor, *Workshop on RFID Security – RFIDsec 2010, 6th Workshop, Istanbul, Turkey, June 7-9, 2010, Proceedings*, volume 6370 of *Lecture Notes in Computer Science*, pages 189–202. Springer, 2010.

[HFW11] Michael Hutter, Martin Feldhofer, and Johannes Wolkerstorfer. A Cryptographic Processor for Low-Resource Devices: Canning ECDSA and AES like Sardines. In Claudio Agostino Ardagna and Jianying Zhou, editors, *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems, Fifth International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011, Proceedings.*, volume 6633 of *Lecture Notes in Computer Science*, pages 144–159. Springer, 2011.

[HMF07] Michael Hutter, Stefan Mangard, and Martin Feldhofer. Power and EM Attacks on Passive 13.56 MHz RFID Devices. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 320–333. Springer, September 2007.

[HMV04] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2004.

[Int95] International Organisation for Standardization (ISO). ISO/IEC 7816-4: Information technology - Identification cards - Integrated circuit(s) cards with contacts - Part 4: Interindustry commands for interchange. Available online at `http://www.iso.org`, 1995.

[Int01] International Organization for Standardization (ISO). ISO/IEC 14443-3: Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part3: Initialization and Anticollision. Available online at `http://www.iso.org`, 2001.

[Int08] International Organization for Standardization (ISO). ISO/IEC 14443-4: Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part4: Transmission Protocol. Available online at `http://www.iso.org`, 2008.

[Int11] International Organisation for Standardization (ISO). ISO/IEC 7816-4: Identification cards – Integrated circuit cards – Part 1: Cards with contacts – Physical characteristics. Available online at `http://www.iso.org`, 2011.

[iS11] RF iT Solution. Rf-it-solution homepage. Online `http://www.rf-it-solutions.com/`, 2011.

[Iva05] Ivanov Aleksey. Side-Channel Attacks. 2005.

[JD02] Vincent Rijmen Joan Daemen. *The Design of Rijndael AES - The Advanced Encryption Standard*. Springer, Berlin, 2002.

[JMV01] Don B. Johnson, Alfred J. Menezes, and Scott Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, August 2001.

[JPS05] Marc Joye, Pascal Paillier, and Berry Schoenmakers. On Second-Order Differential Power Analysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Kob87] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[Koc96] Paul C Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Othzer Systems. 1996.

[Kor10] Thomas Korak. NFC Security Application. pages 1–13, 2010.

[MC11] Mini-Circuits. Mini-circuits. Online `http://www.minicircuits.com/pdfs/ZFL-1000LN+.pdf`, 2011.

[Med07] Marcel Medwed. Template-Based SPA Attacks on 32-bit ECDSA Implementations. Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, December 2007.

[Mes00] Thomas S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

[Nat08] National Institute of Standards and Technology (NIST). FIPS-180-3: Secure Hash Standard, October 2008. Available online at `http://www.itl.nist.gov/fipspubs/`.

[Nat09] National Institute of Standards and Technology (NIST). FIPS-186-3: Digital Signature Standard (DSS), 2009. Available online at `http://www.itl.nist.gov/fipspubs/`.

[NFC07] NFC Forum. NFC Forum Type 4 Tag Operation - Technical Specification, March 2007.

[oSN01] National Institute of Standards and Technology (NIST). Fips-197: Advanced encryption standard (aes). Online `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`, 2001.

[OST05] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache Attacks and Countermeasures : the Case of AES. pages 1–25, 2005.

[Osw03] Elisabeth Oswald. Enhancing Simple Power-Analysis Attacks on Elliptic Curve Cryptosystems. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 82–97. Springer, 2003.

[PHF] Thomas Plos, Michael Hutter, and Martin Feldhofer. Evaluation of Side-Channel Preprocessing Techniques on Cryptographic-Enabled HF and UHF RFID-Tag Prototypes.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. ISSN 0001-0782.

[SM07] Thomas Popp Stefan Mangard, Elisabeth Oswald. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, Berlin, 2007.

[Sti02] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 2nd edition, 2002. ISBN 1-58488-206-9.

[WFF10] Erich Wenger, Martin Feldhofer, and Norbert Felber. Low-Resource Hardware Design of an Elliptic Curve Processor for Contactless Devices. In Yongwha Chung and Moti Yung, editors, *WISA*, volume 6513, pages 92–106. Springer, 2010.

[Xil08] Inc. Xilinx. Xilinx, inc. spartan-3 fpga family data sheet. Online `http://www.xilinx.com`, June 2008.