Master Thesis

Design and Implementation of a Trusted Communication Channel based on NFC

Trebo Fioriello, Manuel, BSc

Institut für Technische Informatik
Technische Universität Graz

Assessor: Ass. Prof. Dipl. -Ing. Dr. techn. Steger Christian
Advisor: Ass. Prof. Dipl. -Ing. Dr. techn. Steger Christian
Dipl.-Ing. Menghin Manuel

Graz, September 2013

Kurzfassung

Durch den einfachen Kommunikationsaufbau zwischen Geräten erfreut sich Near Field Communication (NFC) steigender Beliebtheit. Dabei wird diese Kommunikationsart nicht wie bisher meist nur zum einfachen Lesen von Identifikationsdaten, sondern zunehmend auch zum Steuern von Aktoren und Lesen von Sensordaten verwendet. Bei bestehen Lösungen ist meist nicht der gesamte Kommunikationsweg durch kryptographischen Methoden geschützt weshalb sie nicht für sicherheits-relevante Aufgaben einsetzbar sind. Werden diese dennoch verwendet kann es unter Umständen zum Verlust persönlicher Daten kommen. Dies kann ernste soziale und ökonomische Konsequenzen mit sich bringen. Aus diesen Gründen ist ein sicheres und vertrauenswürdiges System wünschenswert.
Ziel dieser Masterarbeit ist es Kommunikationsmöglichkeiten zwischen NFC-fähigen Geräten zu untersuchen. Dabei wird insbesondere auf sichere und vertrauenswürdige Kommunikation von Anfangspunkt (z.B. Smart Phone) bis Endpunkt (z.B. Sensor) wertgelegt. Dazu wurden zwei verschiedene Varianten implementiert. Bei der ersten Methode werden elliptische Kurven nach Diffie-Hellman zum Schlüsselaustausch verwendet. In der zweiten Methode wurde ein Verfahren zum Schutz vor Zugriff durch unbefugte Personen implementiert, welches auf elliptischen Kurven mit digitalen Signaturen basiert.
Da durch die erweiterten Sicherheitsfunktionen auch der Energieverbrauch steigt, ist es besonders bei batteriebetriebenen Geräten wichtig einen Mittelweg zwischen hoher Sicherheit und Energieverbrauch und somit auch Akkulaufzeit zu finden. Deshalb wird in dieser Masterarbeit auch auf Möglichkeiten zum Senken des Energiebedarfs bei gleichbleibendem, oder sogar steigendem, Sicherheitslevel eingegangen. Weiters werden auch Methoden zur Bestimmung der besten Balance zwischen Energieverbrauch und Sicherheitslevel vorgestellt.

Abstract

Near Field Communication (NFC) rises in popularity as a simple method of establishing communication between devices. This way of communication is not only restricted to the identification of data, it can also be used to read out attached sensors or to control actuators. The main problem of the current implementations is that the complete communication path is not protected by cryptographic methods. Consequently, a loss of personal data, which can create social and economic disadvantages, can occur. For these reasons, a secure and trustful system is highly recommended.
This master thesis will show possibilities to not only protect the communication channel, but also to guarantee a secure path from the source (e.g., smart phone) to the last node (e.g., sensor). Therefore, two different variants are implemented, one uses elliptic curve Diffie-Hellman for key exchange and the other one elliptic curve digital signature algorithm to limit the access to authorized persons.
Additional security needs also more energy. This consumption is an important criteria especially in battery powered embedded systems. Therefore, in order to save energy it is very important that the chosen security level is not too high. Moreover, some possibilities about how to save energy of the target device and at the same time rise the security level of the system are presented. Additionally, it is shown a research on how to find the best balance between energy consumption and security.

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.

....

date (signature)

Acknowledgment

Diese Diplomarbeit wurde im Jahr 2013 am Institut für Technische Informatik an der Technischen Universität Graz durchgeführt.

Danksagung an alle am Institut, speziell Christian Steger, Manuel Menghin, Norbert Druml und Christian Goral von Infineon, die mir mit Rat und Tat Hilfestellungen gegeben haben.

Danksagung an meine Freunde von Südtirol und die neu kennengelernten Freunde während des Studiums, speziell Thomas, Michael und Hannes für die effiziente Zusammenarbeit und den Spass den wir hatten. Ein spezieller Dank geht auch an meine Eltern die mich immer unterstützt und motiviert haben und an Vanesa die jeden Tag für mich da war.

Graz, September 2013 Trebo Fioriello Manuel

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The Near Field Communication business is a fast growing market. IMS research predict, that NFC[1] controller shipment will increase dramatically in the next years and reach from 70 millions[2] up to 900 millions in 2015 [res11]. These predictions show, that NFC will be part of our daily live.

The fast growing NFC market brings also a lot of new applications for NFC devices e.g. from payment to even accessing wireless sensors and actor nodes. Unfortunately, these new applications which use the NFC interface lead to new forms of attacking scenarios and the security has to be tightened to consider this. An example for this issue is the recently solved security leak in Google wallet [RLS12]. This example shows the importance of creating a solid and secure environment before releasing products on the market.

1.1 Motivation

That fast growing market open new possibilities to use NFC in almost every situation. NFC can already be used for more than just read out ID[3]'s from e.g. smart posters. It is possible to control actuators or read out data from sensors in everyday electronic devices like e.g. washing machine or microwave. In the near future it will be used also to read out personal data from e.g. smart meter or to lock or unlock house doors. This use cases will require a higher level of security.

Figure 1.1 shows the concept of device interaction accordingly to Rejhan et al. This master thesis had the goal of implementing a communication from a node (e.g., sensor or actuator) till the smart phone to visualize the data from the node and analyze the system in terms of power consumption using NFC.

[1] Near Field Communication
[2] In year 2011
[3] Identification Data

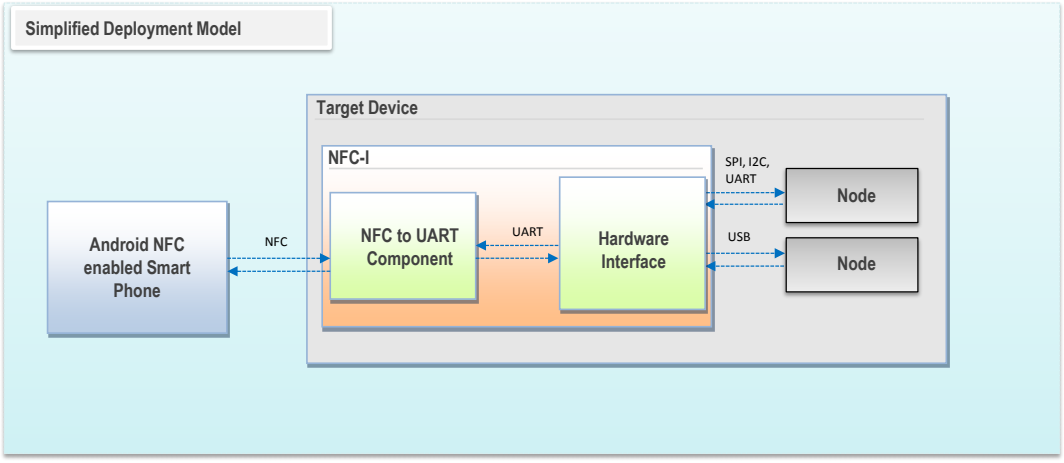


Figure 1.1: Deployment model of the unsecured NFC-Interface by [Rej] et al.

The NFC enabled smart phone establishes a communication with the NFC-I[4] . The NFC-I is basically nothing else than a gateway which sends the received information via serial communication to the hardware interface of the target device. The target device collect the needed information from the nodes and sends it over the hardware interface to the NFC-I which forward this information to the smart phone. Finally the smart phone displays the received information.

So far the communication from smart phone to the NFC-I was without any cryptographic protection, everyone with an antenna and some knowledge about the topic was able to eavesdrop the communication and get all the transmitted data. The data from ID's are not very critical from the perspective of privacy and security. Therefore, there was no necessity to crypt the data. The new opened possibilities to used this technology will require a higher level of trust.

Figure 1.2 shows the modified secure NFC-I with cryptographic support. Due to the fact that the original model has no cryptographic support and consequently leaks very easily personal data. The major difference of the two models is the introduction of a sophisticated security concept. Another difference is that now the NFC-I is not just an interface that forwards information, it has also to provide cryptographic support as well.

[4]Near Field Communication-Interface

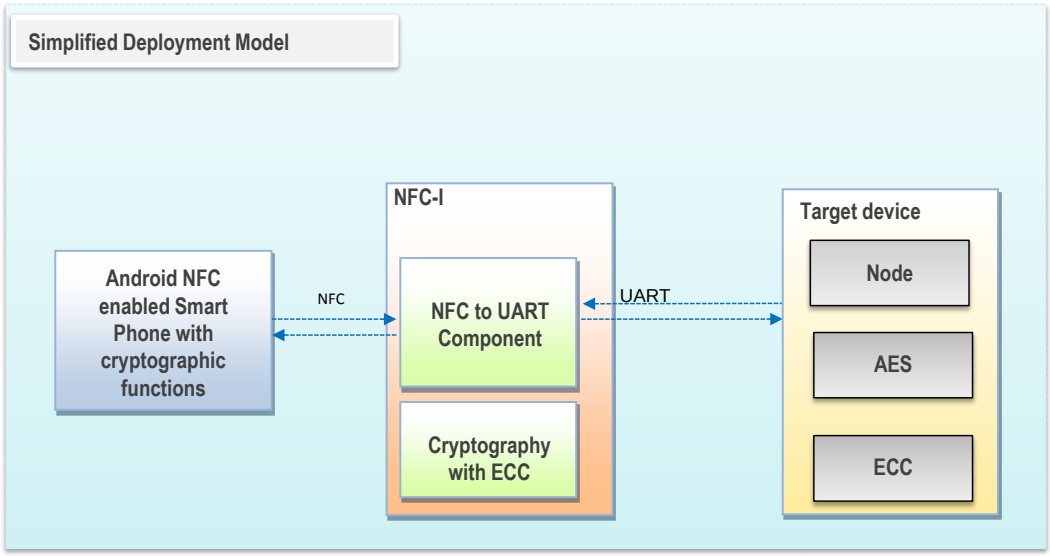


Figure 1.2: Deployment model of the modified secure NFC-Interface.

1.2 Goals

Normally only parts of the system are protected with cryptographic methods, which make it easier for attacker to eavesdrop data on the weak point of the system. This master thesis will show that it is possible to protect the whole system from e.g. a sensor node until the control component e.g. smart phone.

Figure 1.3 shows one of the main goals of this master thesis. The first goal is to implement a secure channel over the whole communication path. Starting from a NFC capable device until a node in the target device. Over the complete communication path an attacker (in our case a sniffer) will not be able to extract valid information from the system. This can only be guaranteed with state of the art cryptography.

A big disadvantage of a system which use cryptography is that the energy consumption increases. This property is important to keep in mind during the design of a battery powered device that should have a long durability e.g. smart meter which is powered by a battery and has to work as long as possible without battery change.

Because of this fact it is important to evaluated different power aware strategies for the system and find the best trade of between security and energy consumption, this defines goal two. Only in this way it is possible to guarantee security and long durability of devices. This is the second important goal which this master thesis will provide.

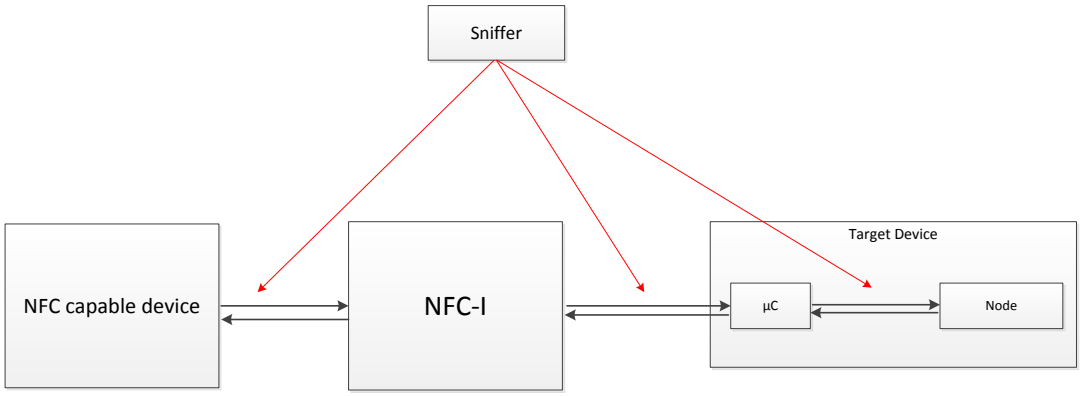


Figure 1.3: Possible attack scenarios on a system.

The third goal is to do a case study on a reference implementation of a smart meter. This case study should include the major two goals and show a way how this trade of between security and energy consumption on this example can be founded.

1.3 Outline

In Chapter 2 an overview of the cryptographic methods used by the industry and their related energy impact is shown. Furthermore, some examples of similar projects are shown and the innovative characteristics of this project are emphasized in comparison with similar ones. In Chapter 3 is shown the design of the architecture of the project. The system is manly subdivided in three communication partners, smart phone side, NFC-I side and FPGA[5] side. Chapter 4 is showns the implementation of the security concept explained in the design chapter. It also shows the complications which showed up during the project and the solution to them. In Chapter 5 and in Chapter 6 is shown a short summary about the problem tasks and the results of this master thesis. Improvement possibilities for further projects are also given.

[5]Field-programmable gate array

Chapter 2

Related work

This chapter provides a short overview about the topic of symmetric, asymmetric cryptography scheme and the related energy consumption. Furthermore it explains how digital certificates and Near Field Communication (NFC) works and finally, similar projects in this field, which are the baseline of this master thesis, are presented.

2.1 Cryptography

Cryptography is a proven method, which has been used along the years and employ secure data exchange between party players. Goals of cryptography are:

1. Confidentiality - The data can only be read by authorized people and cannot be divulged to third parties.
2. Data integrity - The data cannot be manipulated during the transmission.
3. Authentication - The data can only be created by authorized people, devices, artifacts and many more, which ensure that the data is what is claimed to be.
4. Non-repudiation - The data can always be attributed to the creator in order to avoid the negation of previous actions.

Cryptography can be splitted in two big groups: Symmetric and asymmetric cryptography. The difference between them is, that the asymmetric cryptography has a public and a private key and the symmetric cryptography uses only a private key [AJMV01].

Figure 2.1 shows the differences between symmetric and asymmetric cryptography. For the symmetric cryptography a secure channel among the two communication partner is needed for the key exchange. For the asymmetric or public-key cryptography such a secure channel is not needed.

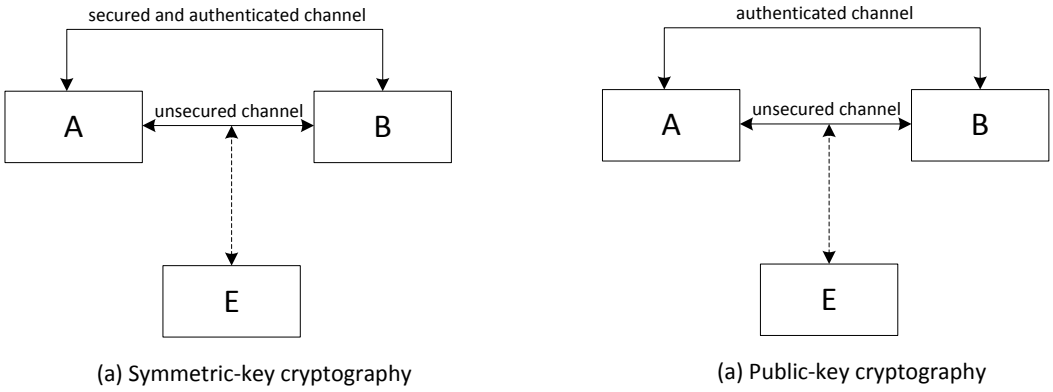


Figure 2.1: Difference between symmetrical and asymmetrical cryptography [HMV03].

2.1.1 Symmetric cryptography

For the symmetric cryptography both players (transmitter and receiver) need the private (secret) key. Only with this private key it is possible to encrypt and decrypt the message.

Algorithm 1 Symmetric decrypt and encrypt of data [DK07]

Summary: decrypt and encrypt process

1. Transmitter (Alice) and receiver (Bob) agree a secret key k.
2. If Alice want to send a message m to Bob, Alice just have to use the encryption algorithm E for encrypting the message $c = E(k,m)$ and sending it to Bob.
3. Bob receives the encrypted message: He uses the decryption algorithm D and the secret key k for decrypt the message $m = D(k,c)$.

The encryption and decryption algorithm are public known, Therefore, they can be used by everyone. However, there is no easily reachable decryption of the message without the secret key.
The most important advantage of the symmetric cryptogrphy is its efficiency in terms of timing and energy consumption, in difference to the asymmetric cryptography. On the other hand, a well-known disadvantage is the exchange of the secret key between both players and the management of the secret key in a network where more players have access. Two symmetric cryptographic standards are DES[1] and AES[2] [DK07].

2.1.1.1 AES

In January 1997, the US National Institute for Standards and Technology (NIST) announced the development of a new standard, the AES. The main idea of the AES was to replace the older standards like DES and triple-DES.
AES is a block cypher, the block length can be chosen 128 *bit*, 192 *bit* or 256 *bit*. The AES

[1] Data Encryption Standard
[2] Advance Encryption Standard

algorithm is also known as the Rijndael algorithm [DR02].

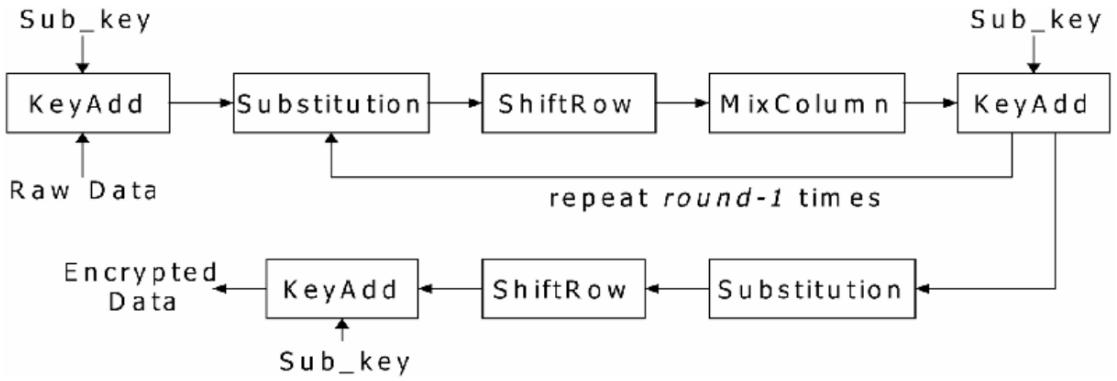


Figure 2.2: Algorithm flow of Rijndeal [KV01].

Figure 2.2 shows the flow of the Rijndeal algorithm for one encryption. The first step (*KeyAdd*) is an XOR operation with the input data and the first round key. The repetition of the next four steps depends on the chosen rounds of the algorithm. The second step (*Substitution*) is the only nonlinear transformation of the hole algorithm, which consists on a byte wise change of the original values with values from a sbox. The third step (*ShiftRow*) is a left shift which depends on the key length and the selected row. The fourth step (*MixColumn*) is a fixed polynomial multiplication with the shifted row. The following steps are the same as described before, the only change is the used round key [DK07].

2.1.2 Asymmetric cryptography

Asymmetric cryptography was introduced in 1975 by three mathematics Diffie, Hellman and Markle. The main idea is to solve the only disadvantage of the symmetric cryptography, the private key exchange. In contrast to the symmetric cryptography, the asymmetric cryptography does not need a secure channel for the key exchange since an authentic channel is sufficient for the exchange of the public key.

Algorithm 2 Asymmetric decryption and encryption of data [HMV03]

Summary: decryption and encryption process

1. Transmitter (Alice) and receiver (Bob) generates both a private key d and a public key e. The keys have to be chosen in an mathematical way, in which the obtaining of the private key from the public key is not possible.
2. Alice want send a secret message m to Bob.
3. For sending the message Alice need the public key e_B from Bob. Only with the public key of Bob Alice can encrypt the message $c = ENCe_B(m)$ and after send it to Bob.
4. Bob use his private key to decrypt the message $m = DECd_B(c)$.

The generated key pairs are based on mathematical problems, which are very difficult to solve if the knowledge about all the properties is not given [HMV03]:

1. The integer factorization problem, which is used for the RSA[3] method.
2. The discrete logarithm problem, which is used for the DSA[4] method.
3. The elliptic curve discrete logarithm problem, which is used for the ECC[5] method.

2.1.2.1 RSA

RSA was developed in 1977 from three mathematics Rivest, Shamir and Adleman. It is the most used encryption scheme because it brings not only security but the possibility to combine it with digital certificates. The security of RSA is based on the mathematic problem of the integer factorization [HMV03], [RSA78].
For security reasons, the key length of RSA nowadays should not be shorter than 1024 *bit* since in 2010 the first integer factorization of 768 *bit* was successfully carried out [BKK+09].

Algorithm 3 RSA key pair generation flow [AJMV01]

Summary: Each player generates a private and a public key

1. Generate two random primes p and q with the same length.
2. Compute $n = p * q$ and $\phi = (p-1) * (q-1)$.
3. Find a random integer number e, $1 < e < \phi$ that $\gcd(e, \phi) = 1$.
4. Use the extended euclidean algorithm to calculate d, $1 < d < \phi$, that $e * d \equiv 1 \pmod{\phi}$.
5. The public key is (e, n), the private key is d.

Algorithm 4 RSA encryption and decryption of data [AJMV01]

Summary: Bob encrypt a message m for Alice and Alice decrypt it

1. Bob receives from Alice the authentic public key (n, e).
2. Represent the message m as an integer in the interval $[0, n-1]$.
3. Compute $c = m^e \bmod n$.
4. Send encrypted message c to Alice.
5. Alice use the private key in order to decrypt $m = c^d \bmod n$ the encrypted message c.

The algorithm 3 is used for the generation of the key pairs (private and public key). Only with this two keys the possibility to encrypt and decrypt messages is given. The algorithm 4 is used to encrypt and decrypt the message.

[3] Developed 1977 from Rivest, Shamir and Adleman
[4] Digital Signature Algorithm
[5] Elliptical Curve Cryptography

2.1.2.2 ECC

The elliptic curve cryptography was proposed in the middle of the year 1980 by Victor S. Miller [Mil86] and Neal Koblitz [Kob87], [BSS99].
The underlying mathematical - elliptic curve discrete logarithm - problem is the most difficult of the three illustrated problems since the calculation of the discrete logarithm of eliptical curves is considered far more difficult than the factorization of integers. Therefore, with ECC one has the possibility of using shorter key lengths, with the same security level that would be obtained in RSA [HMV03].

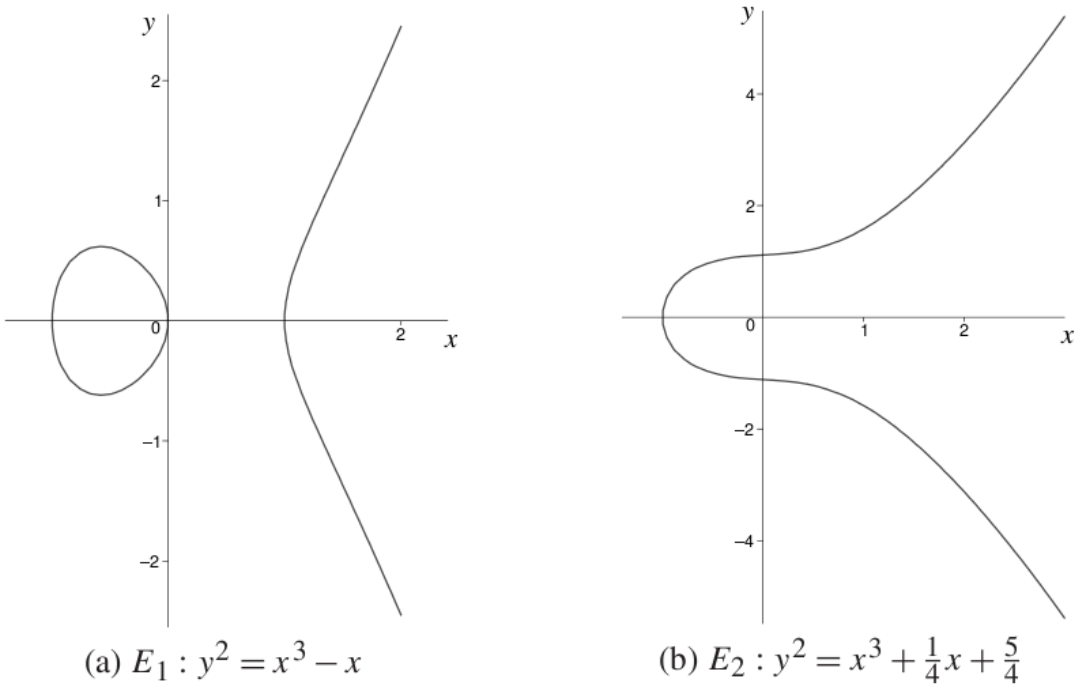


Figure 2.3: Possible elliptic curves over $\mathbb{R}$ [HMV03].

Figure 2.3 illustrates two possible elliptical curves. The form of the curve is very important for the security factor, because not all the curves have the same security level. Therefore, it is important to choose a secure curve. The security level of a curve can be checked through the algorithm of Schoof [Sch85]. In this way it is easy to find out if the used curve is securer than other curves.
There are three forms of elliptical curves:

1. Weierstrass form: $y^2 = x^3 + a_4 * x + a_6$
2. Edwards form: $x^2 + y^2 = 1 + d * x^2 * y^2$
3. Montgomery form: $B * y^2 = x^3 + A * x^2 + x$

A list of secure curves for all the three forms can be found in this paper [ILR].

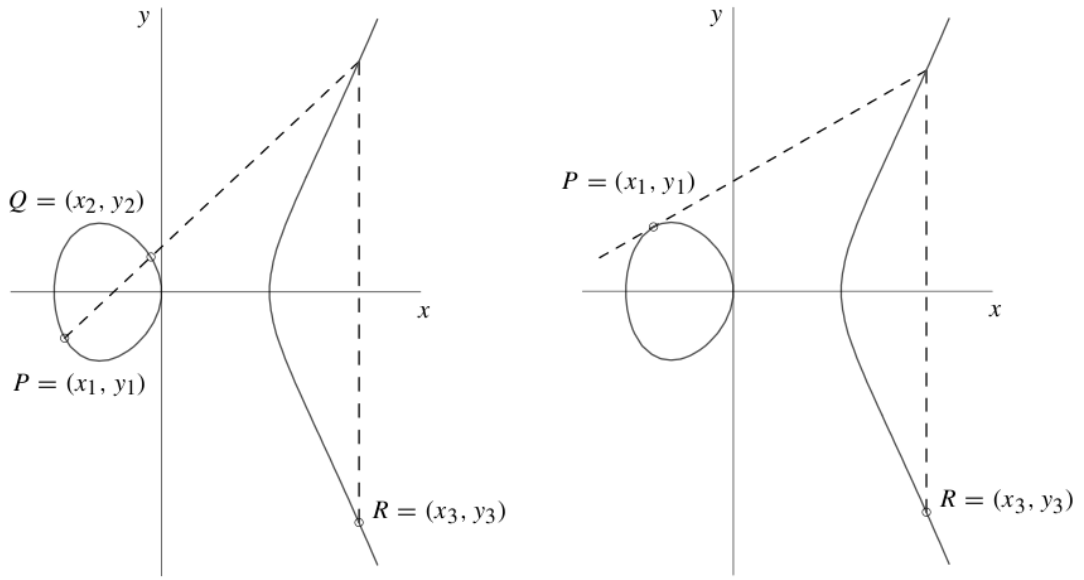
Algorithm 5 shows how the generation of a private and a public key with the Diffie-Hellman method works. It is easy to calculate x^{a*b}, if a and b are known, otherwise the calculation would need a long period of time.

Algorithm 5 ECC - Diffie-Hellman - key pair generation flow [key]

Summary: Bob and Alice generate the public key

1. Alice and Bob choose publicly a cyclic group G and a generator x of G.
2. Alice and Bob generates a random integer, the private key, a and b.
3. Alice compute x^a and Bob compute x^b and they transfer each other the value over an unsecured channel.
4. Now both can compute x^{a*b}, because of $x^{a*b} = (x^a)^b = (x^b)^a$.

Figure 2.4 shows a geometric interpretation of the addition and duplication of points on an elliptic curve. The multiplication and exponentiation in the algorithm 5 can be changed by adding a point or doubling a point on the elliptical curve. A point multiplication $n * P$ is a n times the addition of the point P. The addition of one point with itself (doubling), $R = P + P$, is to interpret like a tangent. The tangent has only one intersection $-R$, Therefore, the point has to be inverted to be obtained R on the elliptical curve. The decrypting and encrypting is based on this principle.



(a) Addition: $P + Q = R$. (b) Doubling: $P + P = R$.

Figure 2.4: Geometric addition and duplication of points on an elliptic curve [HMV03].

2.1.2.3 Comparison of the different cryptographic methods in terms of the security level

Table 2.1 shows the key length of the different methods with the same security level. This comparison provides an evidence: the symmetric cryptography method (AES) needs the shortest key length. For the asymmetric cryptography ECC algorithms need shorter key lengths than RSA and therefore, ECC should be used in new systems. Currently state of the art for ECC security on smartcards are key lengths higher 160 *bit* and for RSA key lengths higher 1024 *bit*.

AES key length (bit)	RSA and Diffie-Hellman key length (bit)	ECC key length (bit)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 2.1: Comparison of the different key lengths to there security level [Ell09].

2.1.3 Digital signature

Digital signatures are used for guaranteeing data integrity, authentication and non-repudiation[6]. Therefore, it exists a public and private key, like for asymmetric cryptography. With these key pairs it is possible to sign and verify the digital signature. The private key is used for signing and the public key for verifying if the message is from the desired sender.
Figure 2.5 shows the flow for digital signatures. Alice wants to sign a message. Therefore, the first step is to create, with the help of a hash-function,[7] a footprint of the message to sign. The second step is to create the signature with the help of Alice private key, the footprint of the message and the signature-function. The third and last step is to add the digital signature to the original message.
For checking the authenticity of people it possible to use certificates. This certificates are equal to identity cards. In 1988 CCITT[8] (nowadays ITU[9] [tOSITD00]) standardized the certificates. Nowadays the standard for smart cards is X.509 in version 3. The verification of this certificates on the smart cards side is not an easy challenge. Therefore, it is done, most of the time, from a connected computer [Wol].
Digital signatures can be combined with ECC. A possible scenario is to verify the trust level of a public accessible terminal (for further information see chapter 2.3.1). It shows how efficient the combination of ECC and digital signature can work. Furthermore it also shows that if the digital signature is created the verification of them is done in a few milliseconds.
Van Damme in [VDWKP09] and chapter 2.3.2 shows how a transfer-flow of electronic

[6] Proof of integrity and origin of data.
[7] Creating a unique checksum of the message, for more details see chapter 2.1.4
[8] Comité Consultatif International Téléphonique et Télégraphique
[9] Telecommunication Standardization Sector

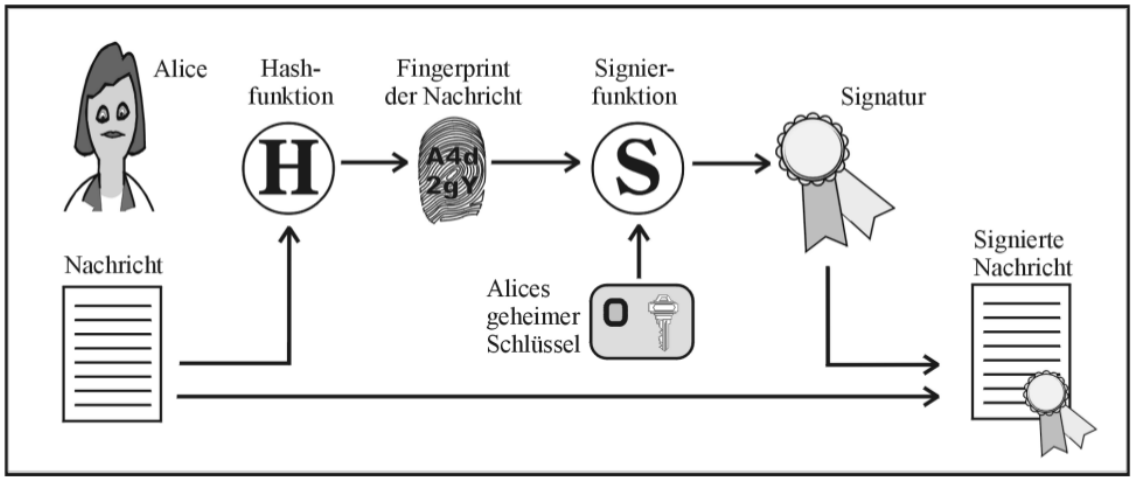


Figure 2.5: Flow to sign messages with digital certificate [Wol].

voucher from phone-to-phone works. Therefore, digital signatures are used to transfer to the recipient the public key of the sender.
Digital signature can also be used in the field of embedded systems. Chapter 2.3.3 shows what is needed therefore. Due to the fact that in the embedded world the power consumption is an important fact, it is necessary to think about how secure and trustful a system has to be. Therefore, a trade of between security and energy consumption has to be founded.
To guarantee the trust of systems up to the sensor, an approach like that one shown in Chapter 2.3.4 can be used. With minimal hardware overhead it is possible to have a trust which starts at the sensor and ends at the desired device.

Figure 2.6 shows, the structure of a $X.509\ v3$ certificate in his full length. The certificate is divided in different sections.

- X.509 version number - Indicates the version number.
- Certificate serial number - Unique number for the certificate.
- Issuer name - Name of the CA[10].
- Certificate validity - Validity of the certificate.
- Subject - Name of the owner of the certificate.
- Subject public key info - Public key for the sign algorithm.
- additional information about the certificate - Information about the used algorithm and so one.

Certificates can be bound on persons, organizations and devices. The issuing of such certificates are e. g.: private companies. This companies have the task of binding the

[10]Certificate Authority

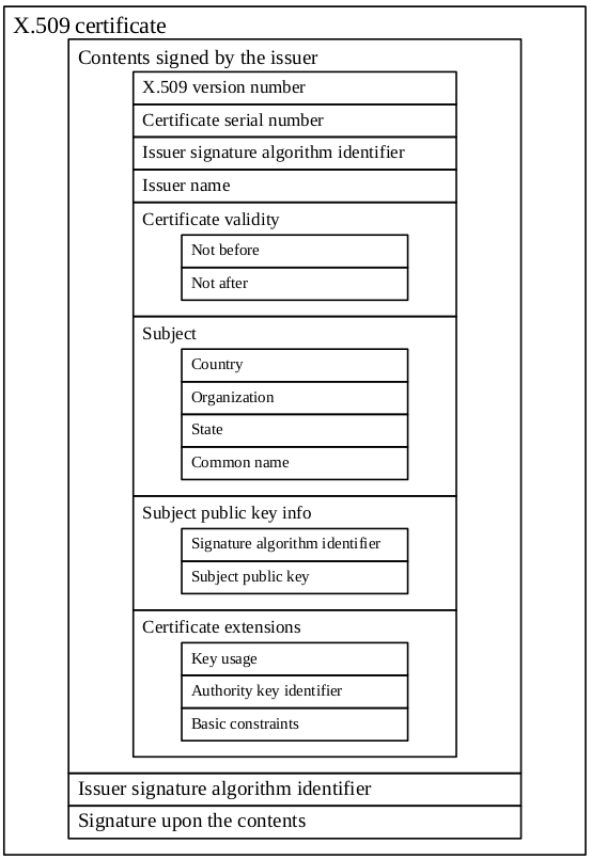


Figure 2.6: Structure of a $X.509$ $v3$ certificate [HLSS06].

public key with information about the owner of this public key.
The flow for owing a certificate is the following:

1. User sends a request to the registration authority (RA). The RA verifies the request and sends it to the certification authority (CA).

2. At the certification authority there are two options: The first one is that the CA generates the public key and the private key. The second one is that the user check the public key and the private key. The CA is not allowed to read the private key under any circumstances. It is possible to give additional information e. g.: restrictions to the certificate.

3. This certificate is sent to the user and is stored in the validation authority (VA). The user should encrypt and store the private key on the hard disc or on a smart card.

4. Over the validation authority everyone has access to the public key for the authentication of messages.

The flow of the public-key infrastructure is shown in Figure 2.7.

Each certification authority offers a blacklist which is accessible over the validation authority. The reasons because of a certificate is marked as invalid are the loss or the compression of the private key.

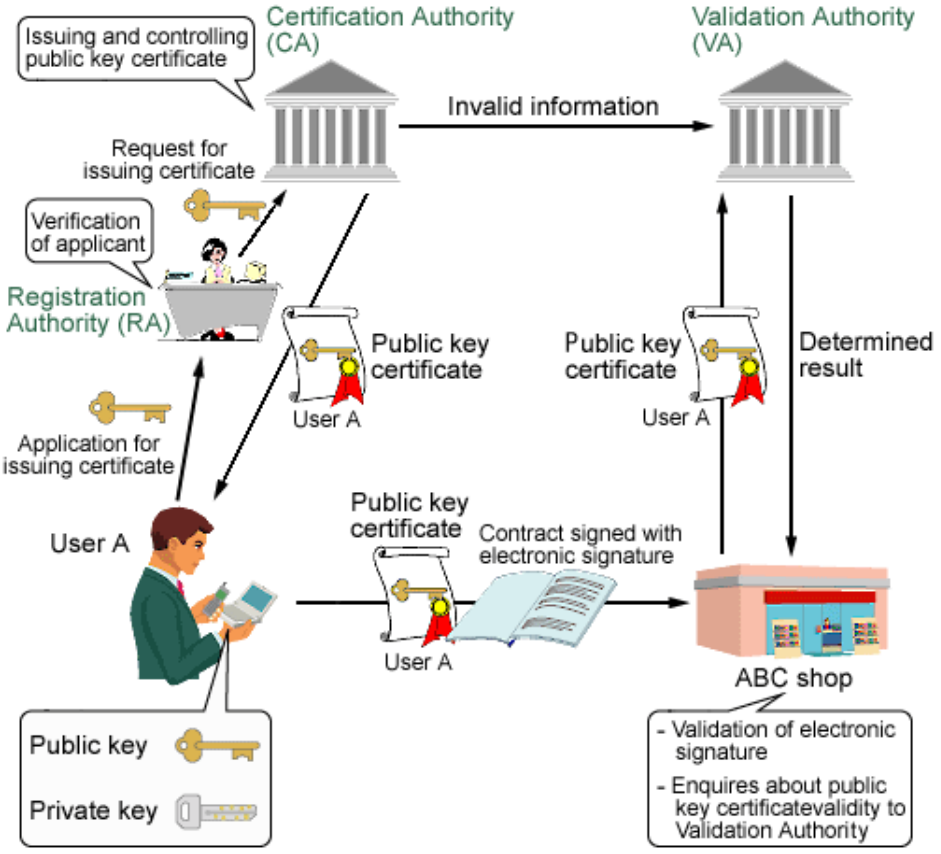


Figure 2.7: Graphical flow of the public-key infrastructure (PKI) [hit].

2.1.4 Hash function

Hash functions are easy to compute and, with them, it is possible to map a large data set to a fixed length. They are used in cryptography to check if there was a modification of the data during the transmission. The most important property of a hash function is that it has to be collision resistant, i. e. two different data sets cannot have the same hash-value [Wol]. Examples for hash functions are MD4[11] [Riv92a], MD5[12] [Riv92b], SHA-1[13] [Sta95] and RIPEMD[14]-160 [DBP96].

[11] Message-Digest Algorithm 4
[12] Message-Digest Algorithm 5
[13] Secure Hash Algorithm - 1
[14] RACE Integrity Primitives Evaluation Message Digest

2.1.4.1 SHA-1

SHA-1 produces as output always a 160-bit long footprint of the input data set (smaller 2^{64} bit). Theoretically it has a security level of 80-bit[15] which can be reduced to a security level of 69-bit by applying appropriated methods[Kil09], [Sta95]. SHA-1 is used by several governments and the industry for security applications like digital signatures [WYY05].

2.1.5 Energy consumption for cryptography on a PDA

Another important factor for choosing the correct cryptographic method is the energy consumption. In particular, in battery-powered systems the energy consumption is a very important fact to keep in mind during the design of an embedded system. Table 2.2, Table 2.3 and Table 2.4 show an overview of symmetric and asymmetric cryptographic methods and the related energy consumption. The measurement configuration for this value can be seen in Figure 2.8. It consists of a secure client server communication through a wireless access point. The client side is composed of an PDA[16] connected through serial port to the power measurement system, which uses LabVIEW for calculating the energy consumption. The different algorithms are executed through the software on the PDA.

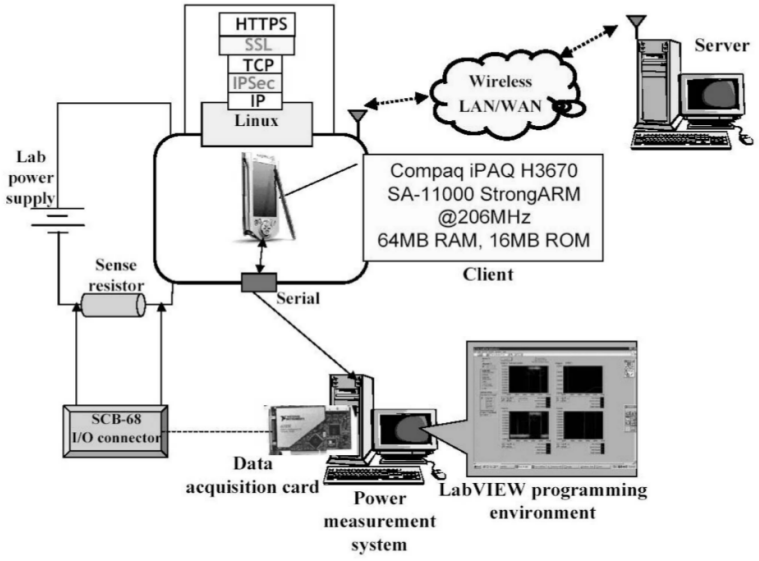


Figure 2.8: Measurement configuration for measuring energy consumption of the PDA [PRRJ06].

[15] In comparison with an symmetrical method.
[16] Personal Digital Assistant

Algorithm	Key size (bit)	Key generation (mJ)	Sign (mJ)	Verify (mJ)
ECDSA	163	226.65	134.20	196.23
ECDSA	193	281.65	166.75	243.84
ECDSA	233	323.30	191.37	279.82

Table 2.2: Energy costs for asymmetric cryptographic systems with digital signature on a PDA [PRRJ06].

Algorithm	Key size (bit)	Key generation (mJ)	Key exchange (mJ)
ECDH	163	276.70	163.5

Table 2.3: Energy costs for asymmetric cryptographic systems without digital signature on a PDA [PRRJ06].

Algorithm	Key size (bit)	Key setup (μJ)	ECB[1] ($\mu J/B$)	CBC[2] ($\mu J/B$)	CFB[3] ($\mu J/B$)	OFB[4] ($\mu J/B$)
AES	128	7.83	1.21	1.62	1.91	1.62

[1] Electronic codebook.
[2] Cipher-block chaining.
[3] Cipher feedback.
[4] Output feedback.

Table 2.4: Energy costs for symmetric cryptographic systems on a PDA [PRRJ06].

2.1.6 Energy consumption for cryptography on a micro controller

Nowadays it is possible to implement cryptography also through the software of an 8-bit micro controller. The energy measurements are, consequently, done on a ATmega128L which is widely used in wireless sensor networks. Table 2.5 shows the significant advantage of ECC over RSA in terms of energy consumption. One can see that for Sign function RSA needs ten times more energy than ECDSA[17]. The Verify function, in contrast with the Sign function, needs less energy for RSA than for ECDSA algorithm. For the key exchange RSA and ECC need more or less the same amount of energy on the client side. However, it one takes into account the server side a big difference happens, since RSA needs again ten times more energy than ECC. This value shows how much energy can be saved by choosing ECC instead of RSA. The executed cryptographic code is written in nesC, C and some parts are assembly optimized.
Table 2.6 shows the energy needed to perform a hash algorithm like SHA-1 and a symmetric cryptography with 128 bit [WGE+05].

[17] Elliptic curve digital signature algorithm

Algorithm	Signature (mJ)		Key Exchange (mJ)	
	Sign	Verify	Client	Server
RSA-1024	304	11.9	15.4	304
ECDSA-160	22.82	45.09[1]	22.3	22.3
ECDSA-224	61.54	121.98[1]	60.4	60.4

[1] For this ECDSA an assumption of two point multiplication numbers is given and therefore, known optimization can be done at costs of memory.

Table 2.5: Energy costs for asymmetric cryptographic systems on micro controller side [WGE+05].

Algorithm	Energy
SHA-1	5.9 $\mu J/Byte$
AES-128 Enc/Dec	1.62/2.69 $\mu J/Byte$

Table 2.6: Energy costs for SHA-1 and AES-128 on micro controller side [WGE+05].

A difference in the energy consumption for cryptography on a PDA and a micro controller can be noticed. One reason for that is that on the PDA is running a operating system with an OpenSSL installed on it[Ope01]. It handles the whole cryptographic flow and has no optimization for this hardware. On the micro controller is running a special optimized version of the OpenSSL flow with some speed ups in assembly.

2.2 NFC - Near Field Communication

NFC is a wireless proximity communication technology, compatible with contact-less smart cards which uses RFID[18]. It allows the transfer of data over short distances[19]. Therefore, it operates in the global available unlicensed radio frequency ism band of 13.56 MHz. The most remarkable advantage of NFC is the easy communication technology without paring and a high trust through small transmission ranges. Transactions are initialized automatically by touching a reader or an other NFC device and it allows communication by simply going into transmission range. Estimation shows, that in 2012 about 180 million mobile devices will use this technology. Some of the application fields of NFC are ticketing, virtual coupons and product informations [MLKS08].
There is a big amount of tags and smart cards with different memory sizes. An overview is given in the Table 2.2.

[18] Radio-frequency identification
[19] Up to 10 cm.

Tags / smart cards	memory size
Mifare Ultralight	48 Byte
Innovision Topaz	96 Byte
Innovision Jewel	96 Byte
Mifare Ultralight C	192 Byte
Mifare Mini	320 Byte
Sony FeliCa (RC-S890 IC-token)	496 Byte
Mifare 1k	720 Byte
Sony FeliCa (RC-S885 card)	2464 Byte
Sony FeliCa (RC-S860 card)	2464 Byte
Mifare 4k	3480 Byte
Mifare Plus	3480 Byte
Mifare DESFire	2 kB, 4 kB
Sony FeliCa (RC-S880 card)	6400 Byte
Mifare DESFire EV1	2 kB, 4 kB, 8 kB
Infineon SLE 66CL(X)xxPE	4 kB up to 128 kB

Table 2.7: Overview of tags and smart cards with the relative memory size [Kil09], [Inf11].

2.2.1 Operating modes

NFC possess different operating modes [MLKS08]:

read / write mode In this mode, the data can be read or written from a passive[20] tag. An example is a smart poster[21]

Card emulation In this mode, NFC devices can emulate smart cards in such a way that the reader is not able to see a difference. An example is payment.

Peer-to-Peer In this mode, two NFC devices can establish a bidirectional connection. An example is the exchange of contacts.

Table 2.8 shows in detail the communication flow with the related modes, the communication interface and a use case.

[20] Tag without battery.
[21] The user get more information by reading the NFC tag.

Communication flow	Operation mode	Communication interface	Use case
(1) *Use of unique ID:* NFC device provides the data and the reader saves them	Tag emulation, Read / Write	ISO 14443	Access control
(2) *External mode of secure element*[1]*:* NFC device provides the data and the reader saves it	Tag emulation, Read / Write	ISO 14443	Access control, Payment
(3) *NFC devices read external tag:* Tag provides the data and the NFC device read it	Tag emulation, Read / Write	ISO 14443	WiFi-Config, smart poster
(4) *Data exchange over NFC:* NFC device provides the data and another NFC device reads it	Peer-to-Peer	ISO 18092	WiFi-Config, data exchange
(5) *Internal mode of secure element:* Communication between secure element in NFC device and host controller application	Internal mode, Communication channel to secure element	ISO 7816	Ticketing, Money transfer

[1] Separated chip which contains a secure storage and a secure processor where security critical application are executed.

Table 2.8: Overview of NFC operating modes [MLKS08].

2.2.2 Threats in NFC systems

There are many attack possibilities on NFC systems. Considering the knowledge of how they are made and how they work, it is possible to make the system more secure against attacks. Only in this way, the probability of avoiding the leaking of the private data increases [HB06].

Eavesdropping: NFC uses a contact-less communication interface. Therefore, eavesdropping is an important issue. When two NFC devices transmit data to each other, an attacker may eavesdrop the data with an antenna. Therefore, all the information is available on the Internet. The distance until the range where eavesdropping is possible depends on the operating mode of the sending device. It is possible to eavesdrop

up to 10 meter if the sending device is in active mode[22]. In passive mode[23] it is possible to eavesdrop up to 1 meter.
A solution to this problem is to establish a secure channel between the devices. In this way the eavesdropper can not use the eavesdropped data [HB06].

Data Corruption: The attacker generate an RF field to disturb the communication between the two devices. The benefit of this attack is not the modification of the transmitted data but a denial of service attack.
A solution for this problem is that the sending device controls the RF field after it finishes the transmission. The power which is needed to disturb the transmission is significant higher than the normal used power. In this way, the attack can be detected, but there is no way to stop this attack [HB06].

Data Insertion: The attacker try to answer, with a manipulated answer, to the sending device. This is only possible in case the answering device takes very long period of time to answer. During this long time, the attacker can send this manipulated answer to the sending device. If both messages overlap each other, the data will be corrupted.
There are two solutions for this problem. The first one is not having a time delay until the receiver device returns an answer. In this way, the attacker has no chance to send his manipulated answer. The second one is to establish a secure channel between the two devices [HB06].

Data Modification: The attacker modifies the data before the receiver gets it. From the point of view of the receiver, it is dealing with valid data. However it was modified by the attacker. The feasibility of this method is strongly related to the used amplitude modulation, because the decoding of the signal is different for 100 % and 10 % modulation.
There are two solutions for this problem. The first one is to control the RF field during the transmission and stop it if an anomaly is detected. The second one is to establish a secure channel between the two devices [HB06].

With this high possibility of threats, it is important to use the cryptographic methods which are explained in chapter 2.1 in order to increase the possibility of handling the problems regarding to the security of the system.

[22] Device has his own power supply, i. e.: battery and generates the RF field.
[23] Device does not have his own power supply and the other device generates the RF field.

2.3 Similar works

The next four example will show how to deal with the challenge of having untrusted components in the systems and reading sensors.

2.3.1 A Trusted Platform Module for Near Field Communication

The main goal of this project is the verification of the security state of public accessible terminals. Therefore, a NFC interface and a TPM[24] will be added to the public terminal. The most important advantage that this project provides is that it is a single chip solution i.e. NFC interface and TPM are on the same chip. For the encryption ECC will be used to increase the computation and communication performance.
Figure 2.9 shows the flow of the protocol. The first step is, that the NFC phone send to a public terminal a challenge configuration N_0. The second step is that the public terminal signs the challenge with ECDSA and sends this signed challenge back to the phone. After this the phone can check the security state of the terminal.
The public terminal needs about 2.57 seconds at 13.56 MHz for the generation of the digital signature. The transmission over NFC takes about 22 ms and the verification on the phone about 33 ms. The trust state of a public terminal can be checked in less than 3 seconds [HT11], [HT10].

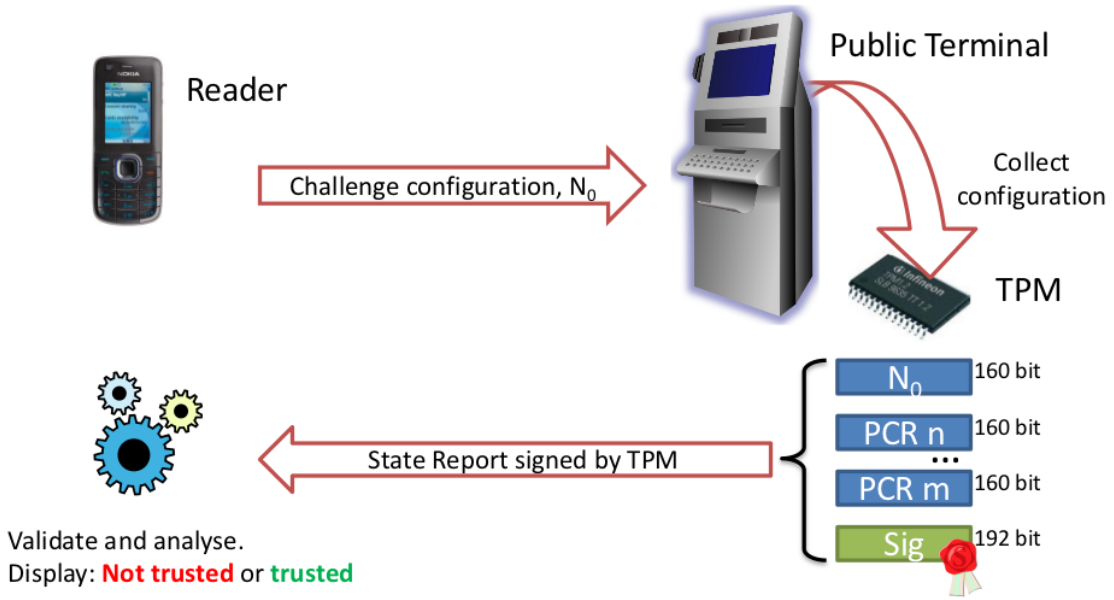


Figure 2.9: Flow of the protocol for a trusted platform module [HT11], [HT10].

2.3.2 Offline NFC Payments with Electronic Vouchers

The introduction of electronic vouchers helps to have a better management over the coupons in circulation. Another advantage is that electronic vouchers are more difficult

[24] Trusted Platform Module

to copy as the coupons made of paper. For electronic voucher only a NFC able phone is needed to use them. The vouchers can be transmitted over SMS[25] or a NFC enabled smart phone.

The electronic voucher system is as follow (Figure 2.10) [VDWKP09]:

1. The customer receives a coupon over SMS or he scan it from a smart poster.
2. The customer can transmit his vouchers to other devices, either by SMS or over NFC.
3. The customer can trade the voucher in a shop.
4. The voucher will be checked if it is valid.
5. The client can easily manage vouchers and the validity of the voucher to use.

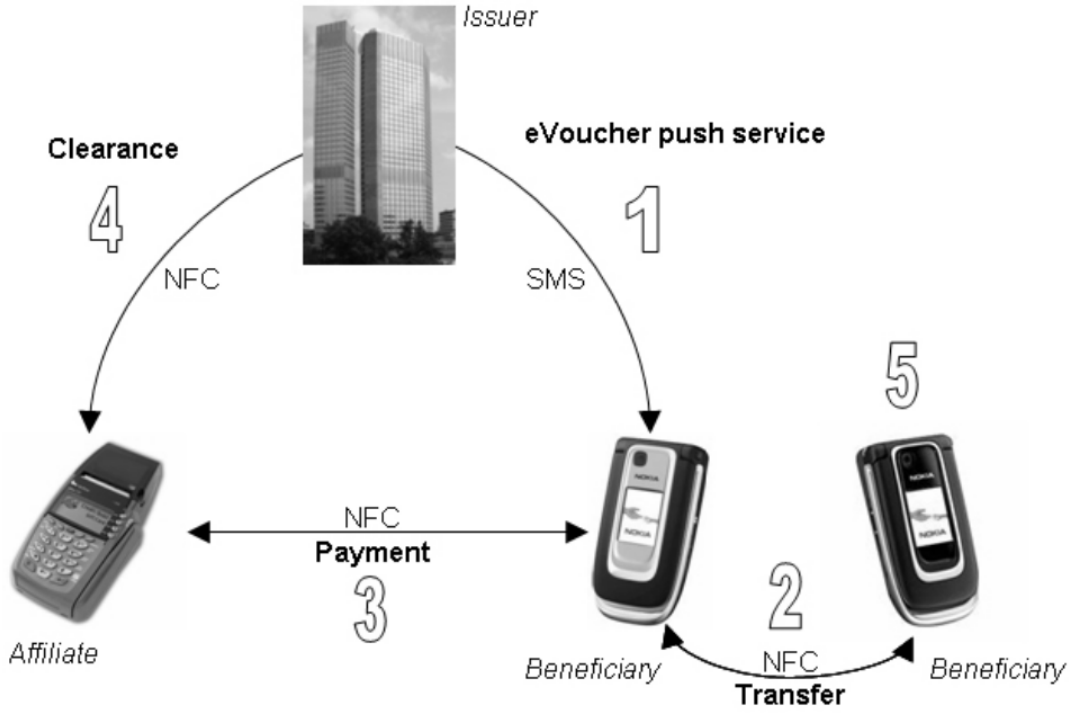


Figure 2.10: Electronic voucher system based on NFC technology [VDWKP09].

The most important issue is that a certain level of security is guaranteed and, specially, that no money get lost during the transmission. The whole voucher management system is based on the phone side. Therefore, the software is very important. The software consists of two parts, the MIDlet[26] and a Java Card application which runs on the secure element.

[25] Short Message Service

[26] A Java ME program written for MIDP (Mobile Information Device Profile). MIDP is a port of the Java platform and is used especially for smart phones or PDAs.

The secure element helps to accept only authorized software.
Figure 2.11 shows the protocol for transferring voucher from phone-to-phone [VDWKP09].

1. The sender starts MIDlet for the transfer of the voucher. The transfer is protected by a PIN[27].

2. The secure element, at send side, start the transfer to the secure element at receive side.

3. The receive generates a random challenge C and send it together with the digital certificate to the sender.

4. The sender verify the digital certificate and generate a random symmetrical key $K1$. This key $K1$ is encrypted with RSA and the resulting output is $R1$. The sender generates a RSA signature as well $R2$ and transfer the encrypted key $R1$ and the digital signature $R2$ to the receiver.

5. The receiver decrypt $R1$ with the help of the private key and get in this way the private key $K1$ from sender, furthermore it verifies the digital certificate. The receiver creates a footprint of the transfered key $K1$ and transfer it as $R3$ to the sender.

6. The sender verifies the footprint of the key and transfer the encrypted voucher $M1$ to the receiver.

7. The receiver decrypt the voucher $M1$ and save him. To guarantee that no data got lost during the transmission. The receiver creates a checksum $M2$ from the transfered voucher and transfer it to the sender.

8. The sender verifies the checksum and delete the voucher.

9. The receiver adds the new voucher in its management system.

There is a large range of similar papers over this topic, other examples are [ADF07] and [DA07].

2.3.3 Plug-n-Trust: Practical Trusted Sensing for mHealth

Early information about the health state of patients can help to improve healthcare quality by improving the efficiency and reducing cost. Especially mobile sensors which are distributed over the body of the patient can help to improve the healthcare. This sensors collect inherently sensitive data of the patient. It is important that the access to them is restricted, otherwise this data can cause social or economic problems to the patient. Therefore, this devices have to be secure and trusted.
Figure 2.12 shows the system model of such a mobile device to control the healthiness of a patient. This device has several sensors distributed over the body of the patient. The communication from the sensors to the mobile device is done over WBAN[28], i. e. Bluetooth or Zigbee. The mobile device can be infected from malicious software. Therefore, it is

[27] Personal identification number
[28] Wireless Body Area Network

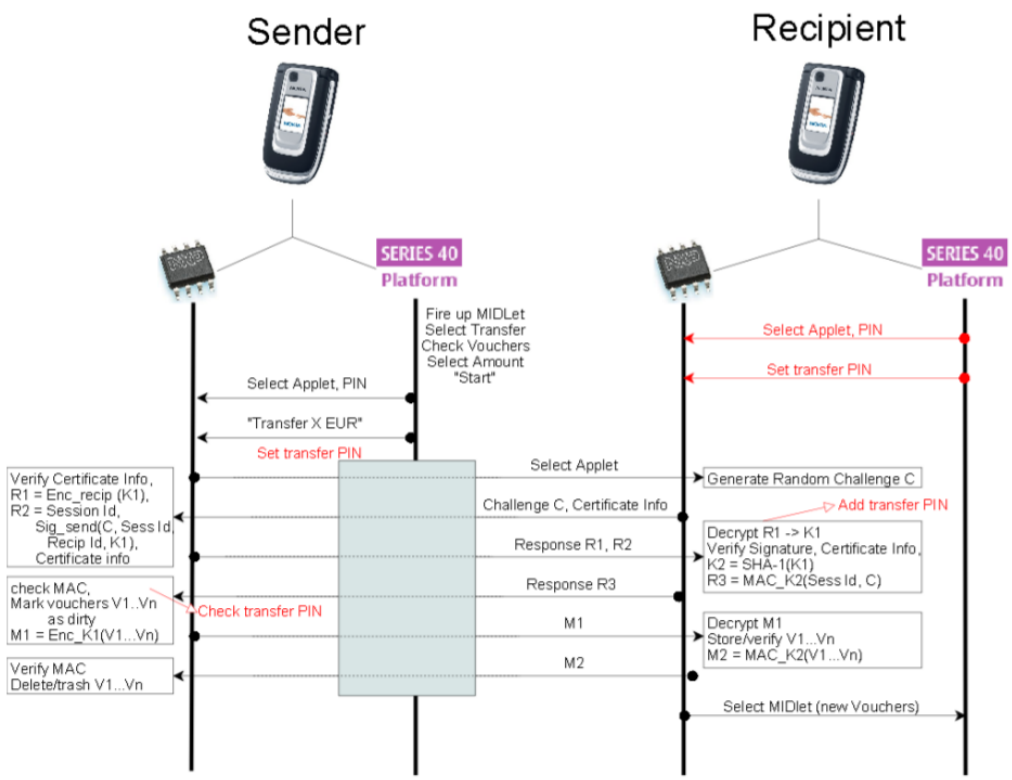


Figure 2.11: Phone-to-phone protocol [VDWP09].

important to guarantee security even when the device is infected. This is done by a special SD[29] card. The collected information is preprocessed on the mobile device and afterwards sent over Internet to an central health recording system, which can be at patient's home or in a hospital [SSPK12].

The hardware have to fulfill following criteria [SSPK12]:

- *Cryptography:* Each sensor note (SN) has to crypt and hash the data before sending them to the mobile device. In order to achieve that, cryptography standard algorithms like AES and SHA are used. This standards are nowadays available in almost every microcontroller.
- *Clock:* Each SN has to have an embedded real time clock, which is needed for the timestamp.
- *Platform:* The mobile device (MN) needs at least two wireless network interfaces, one for the WBAN communication to the sensor nodes and the other for the communication to the Internet for data transfer to the back-end service.

[29] Secure Digital

- *Internal sensors:* The integrated sensors of the mobile device i. e.: camera, GPS[30], clock ... can not be used because they can be compromised.

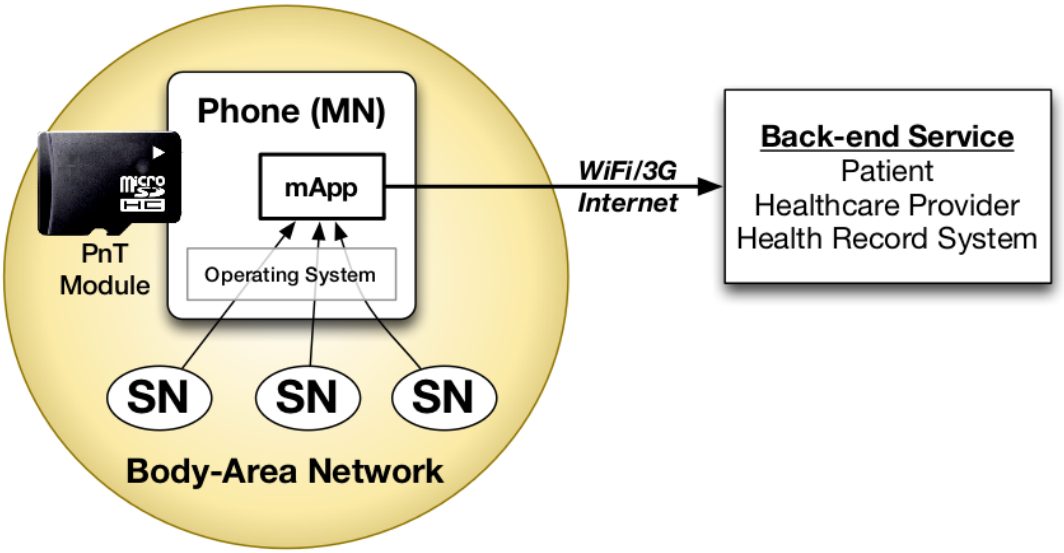


Figure 2.12: System model of the communication from the sensor node to the back-end service [SSPK12].

Because of the property that the mobile device is powered by a battery, it is important the energy consumption of the system. The special SD card consume in idle mode around 9 mW, this value is for one magnitude higher than a normal SD card. In total, for data preprocessing and communication, around 20 % of the daily battery is consumed[31] for the health system [SSPK12].

2.3.4 Trusted Sensors and Remote Sensing

Trusted sensors are important in a high number of applications i. e.: When a house owner goes on holidays and wants to be aware of the state of the electronic devices at home he wants to be provided with trusted information.
Therefore, three components are needed:

1. The usage of public physical unclonable function (PPUF) [BP09]).
2. The usage of random challenges.
3. To use interleaver to ensure trusted information flow.

Public physical unclonable functions (Figure 2.13) are a complex system with a large number of inputs and outputs. In this way, it is guaranteed that the mapping from the

[30] Global Positioning System
[31] Tested on an G1 and Nexus One smartphone.

inputs to the outputs cannot be predicted in any reasonable time. Therefore, the PPUF consists of two parts, a booster (B) and a repressor (R) cells. With the help of booster cells it is possible to increase the switching frequency from output with relation to the input. Repressor cells are used to decrease the switching frequency from input with relation to the output.

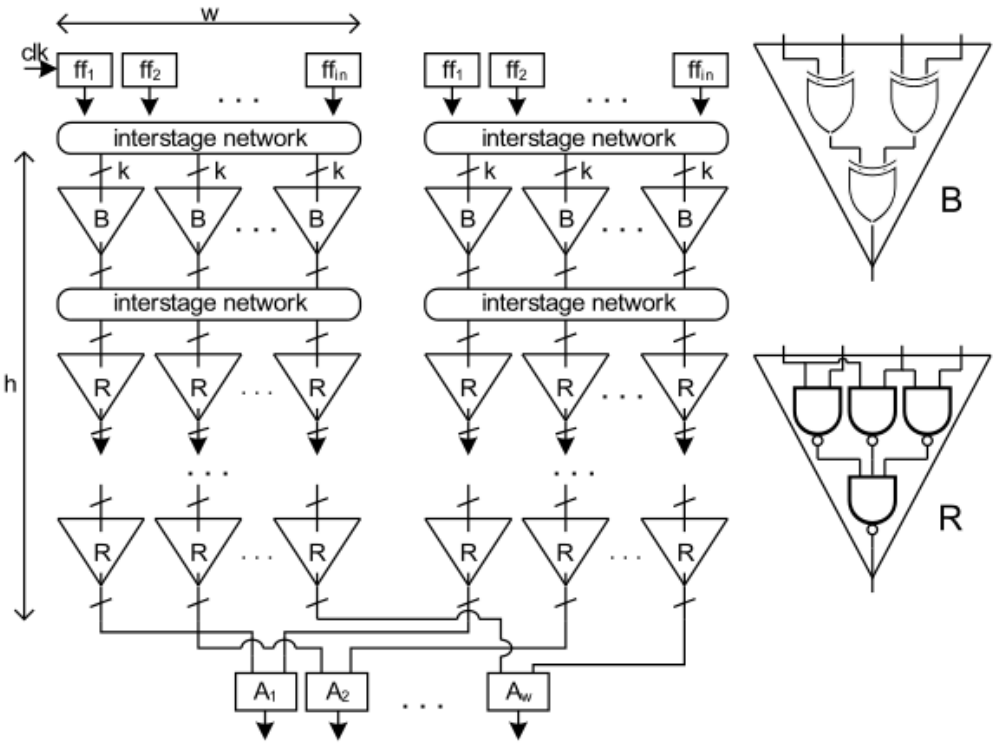


Figure 2.13: Public physical unclonable functions architecture [PMW10].

The trusted sensing system consists of standard cells, two PPUFs and a random challenge generator (Figure 2.14). The random challenge generator gets a key as input and generates a pseudo-random bit string as output. This generated bit string is sent with the sensor data to an XOR. The first PPUF (PPUF Result) is used to authenticate the sensor data. The second PPUF (PPUF System) is used to authenticate the time and location of the sensor data.
The most recognized characteristic of this method is that it works with minimal hardware overhead, low latency and minimal energy consumption [PMW10].

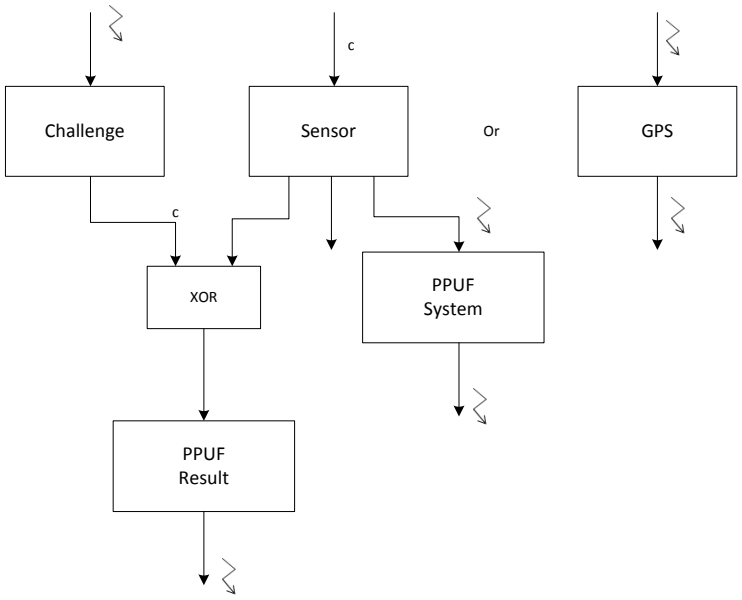


Figure 2.14: Trusted sensing system [PMW10].

Chapter 3

Design

This chapter will show how the design of the system is done. The system can be divided in different components:

- Smart phone
- Communication between the smart phone and the NFC-I
- NFC-I
- Communication between the NFC-I and the target device
- Target device

Figure 3.1 shows the design overview of the system.

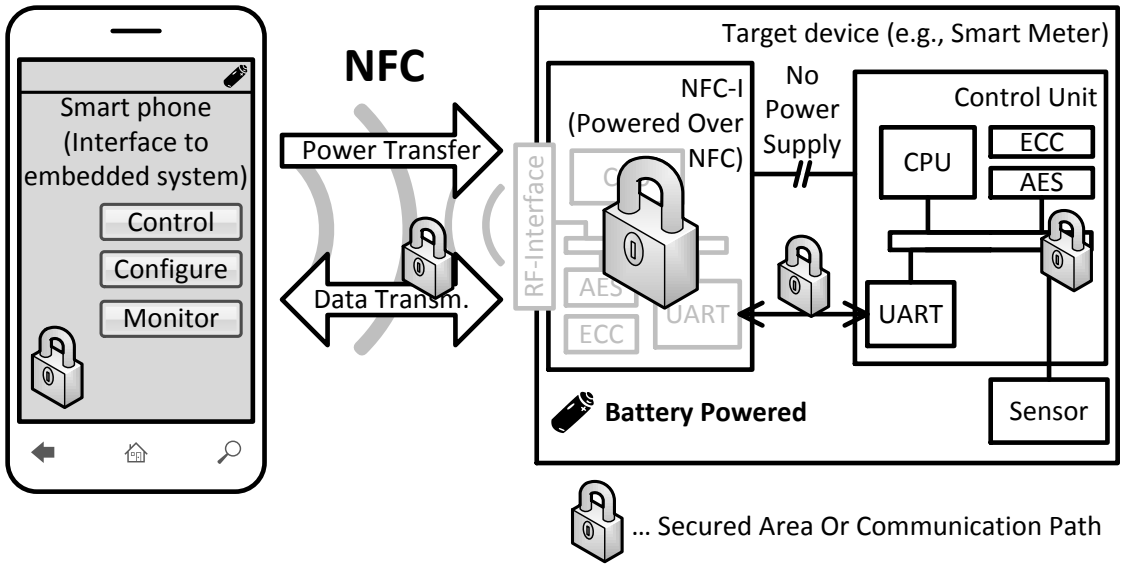


Figure 3.1: Complete design overview, adapted from [MM].

To protect the system against attackers, different cryptographic methods like those showed in the related work are possible. Not all of them are suitable nowadays, because of energy consumption and security.

The decision of which cryptographic type (symmetric or asymmetric) to take depends on the advantages and disadvantages of them. The symmetrical cryptography has the big advantage to use very short key length and at the same time also a high security level, but the disadvantage of the key exchange. Both sides have to know the same key otherwise the cryptography fails. The asymmetric cryptography, in contrast to the symmetric one, uses longer key lengths for the same security level but brings a solution for the key exchange e. g.: Diffie-Hellman. That is the reason by using asymmetrical cryptography for the key exchange and symmetrical cryptography for the data transfer.

For the asymmetric cryptography two different types are available, ECC and RSA. In this master thesis, elliptical curve cryptography will be used, because of the shorter key length compared with RSA (see Table 2.1). This shorter key length reduce the energy consumption by the same security level. For the key exchange ECDH[1] and for the digital signatures ECDSA are used.

For the symmetric cryptography also two types are available, AES and DES. Nowadays DES is not used any more, because of its short key lengths. It has been replaced with AES. Therefore, AES is used in this master thesis.

For the support of ECDSA a hash algorithm is needed. There is a huge amount of different algorithms with different security levels and possible collision attacks. The choice of which algorithm at the end is between $MD5$, $SHA-1$ and $SHA-2$. All of this three hash algorithms are often used and well known. The reason why in this master thesis $MD5$ is not used is, that $MD5$ has been completely broken with collision attack. Therefore, using it is not recommended any more. $SHA-1$ and $SHA-2$ are both securer then $MD5$, but the reason for choosing $SHA-1$ is because it is more common than $SHA-2$. $SHA-1$ has only one disadvantage, that it has been partially broken, it is possible to reduce the normal security level to a minimum of 69 (for further information see section 2.1.4). $SHA-2$ has not been broken yet, but for this master theses the security of $SHA-1$ is sufficient.

Smart phone: The smart phone is used to interact with the system, this includes the graphical interface for the user and the communication to the target device. Over the graphical interface the user can request data from the target device. This data has to be encrypted before being sent to the target device with the showed cryptographic methods.

Communication channel between smart phone and NFC-I: From the smart phone to the NFC-I, a trusted communication over NFC has to be established. Therefore, ECC will be used. This communication is used for two different purposes, bringing the AES key to the target device and forwarding the encrypted message in both directions to the target device and vice versa.

[1] Elliptic curve Diffie–Hellman

NFC-I: The NFC-I is the connection point between the contactless communication over NFC to the smart phone and the contact base communication with the target device. It can operate in two different modes, variant 1 (Figure 3.8) where the NFC-I agree two shared secrets, one with the smart phone and one with the target device and variant 2 (Figure 3.9) where the NFC-I is like a gateway and just forward data in both directions.

Communication channel between NFC-I and Target device: From the NFC-I to the target device, a secure communication has to be established. Therefore, a serial communication will be used (Figure 3.1).

Target device: On the target device the same cryptographic methods as on the smart phone has to be used. Furthermore, the target device has to simulate a smart meter, which has to be secured.

3.1 Smart phone (Interface to embedded system)

The smart phone in this master thesis is used to communicate with the target device. Therefore, the smart phone has to provide a NFC interface. Over this NFC interface the smart phone can access the target device and exchange APDU[2]s with them.

Figure 3.2 shows the flow of the program on the smart phone with the adaption made during this master thesis. By starting the application on the android smart phone, it starts to initialize the application. If the smart phone detects a communication partner it switches to the home screen, which shows all the needed information about the sensor and possibilities to change by touching on the buttons configuration. If a data request from the smart phone to the target device is send, a APDU message is generated and send over the NFC to the NFC-I and afterwards to the target device. The target device evaluate the received request and sends the requested information back. This data is displayed on the main screen and the program is ready to send and receive further requests.

[2]Application Protocol Data Unit

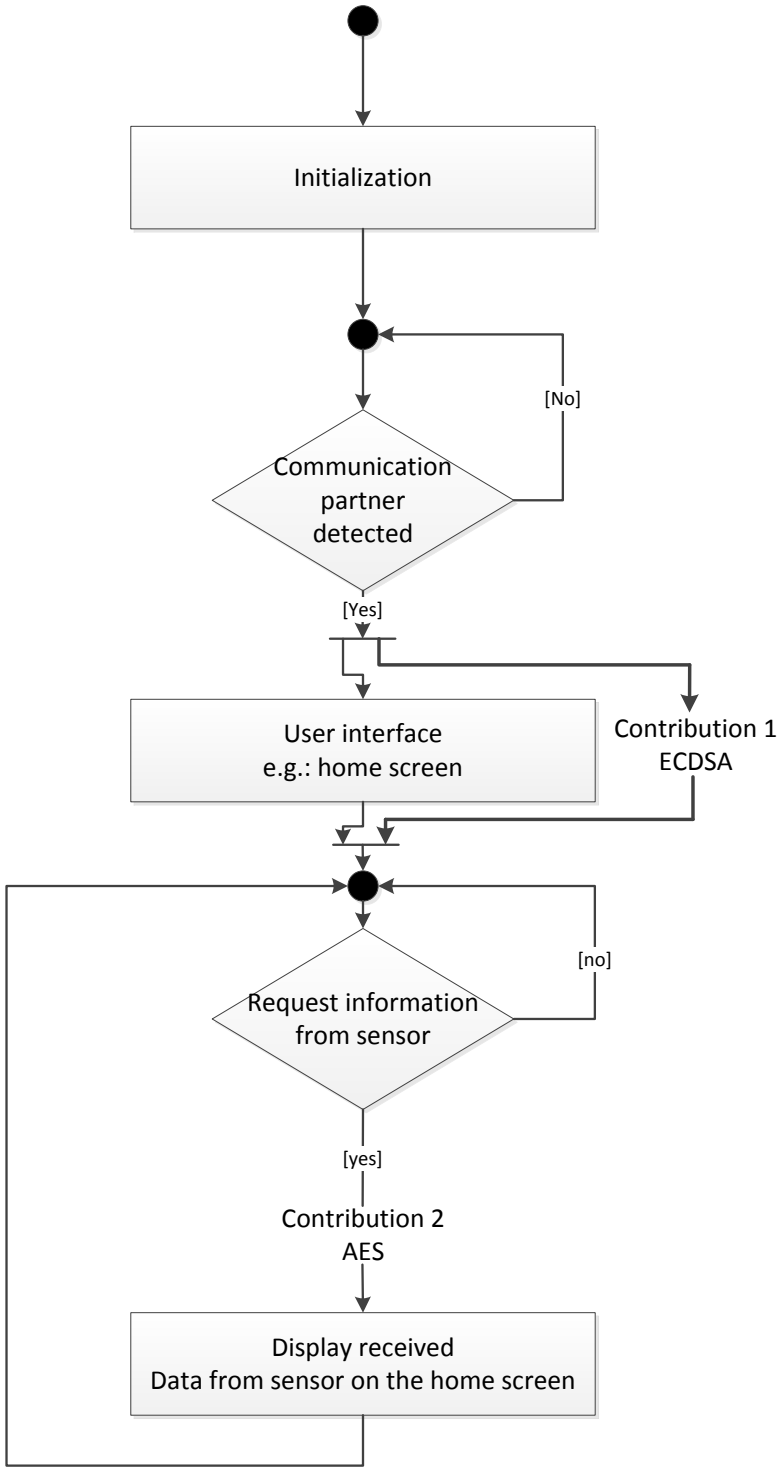


Figure 3.2: Modified program flow overview of the smart phone by [Rej] et al.

3.1.1 Cryptography

Figure 3.3 shows the first relevant security enhancement on smart phone side, of this master thesis. The introduction of digital certificates and cryptography is an important step to secure the communication against attackers and not authorized persons.
For the key generation a cryptographic library will be used. The key generation is needed to generate a public and private key. The public key can be sent without any security restriction over NFC. Moreover the information that potential attackers could eavesdrop has no useful information. The private key is needed afterwards for the shared secret calculation with ECDH. To protect the system against unauthorized persons, the public key has to be hashed and signed before sending it over NFC to the communication partner. If the verification, on the communication partner side was valid, then it starts with the key exchange.

Figure 3.4 shows the second relevant security enhancement on the smart phone side. To guarantee that the transfered information cannot be understood from attackers, the data has to be encrypted before sending them and decrypted when received from the communication partner.
The encryption of the data is done with AES and the used key is the shared secret which was agreed in Contribution 1. The received answer has to be decrypted and forwarded for post processing.

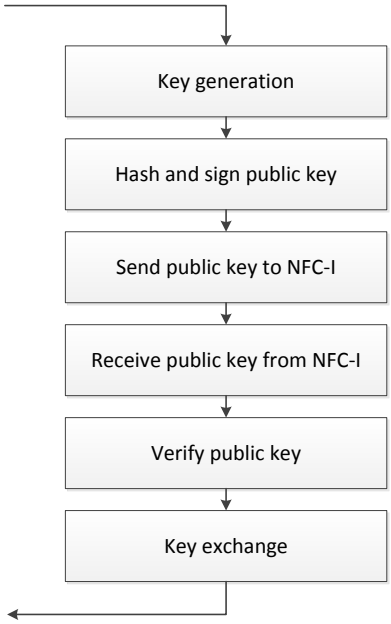


Figure 3.3: Program flow of the smart phone - Contribution 1 - elliptical curve digital signature algorithm and key exchange.

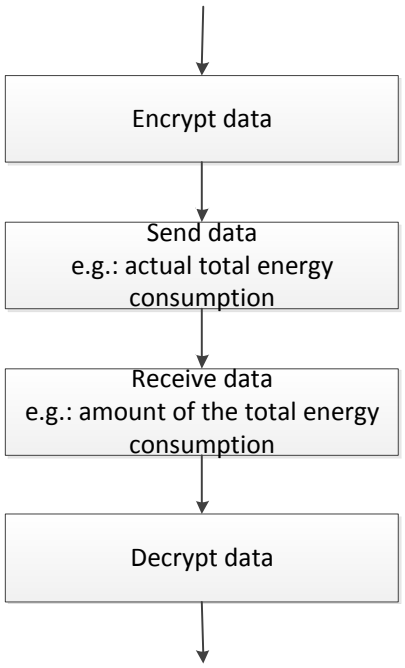


Figure 3.4: Program flow of the smart phone - Contribution 2 - advanced encryption standard for data protection.

3.1.2 Class diagram

Figure 3.5 shows the class diagram on the smart phone side. Over the user application it is possible to interact with the user over a user interface consisting of buttons and message fields. This user application has access to all the needed libraries. For example the user application uses the cryptographic library to generate key pair for the ECC or to calculate the shared secret. Furthermore, the user application can access the base communication library which use the NFC-Module to send and receive data over NFC. The user application gives the desired bytes to be sent to the base communication library, which prepares the message to be sent over NFC depending on the chosen configuration in the user application. After the base communication library has prepared the message, the NFC-Module send it to the NFC-I and waits for its answer, which is sent to the user application for post processing.

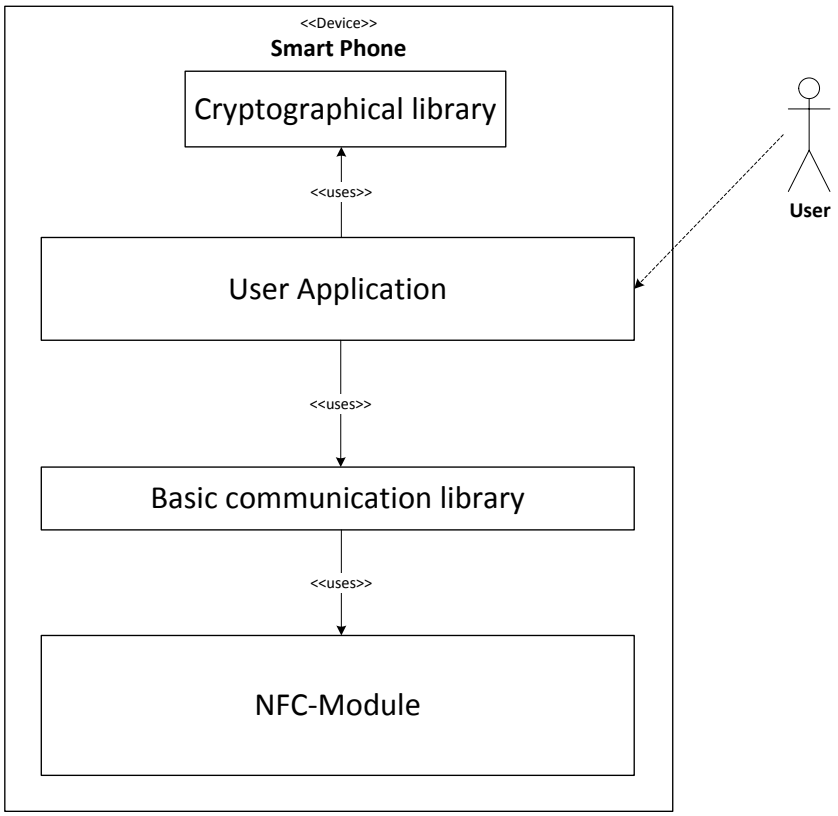


Figure 3.5: Class diagram, smart phone side adapted from Rehjan at all [Rej].

3.2 Communication channel between smart phone and NFC-I

The communication from the smart phone to the NFC-I is only possible over contactless communication. If the NFC of the smart phone is activated and the smart phone get close to the NFC-I, then it get powered with the RF[3]-field of the smart phone. The NFC-I needs no additionally battery or power supply. Over this RF-field also the data is transfered.

To protect the communication, the introduction of cryptography is needed. There are two possibilities which are possible for the asymmetric cryptography. As mentioned before ECDH and ECDSA is used by the smart phone to secure the communication to the target device.

3.2.1 Cryptography - ECDH

Figure 3.6 shows how a Diffie-Hellman key exchange with the NFC-I is done. The following steps describe the exact flow:

[3]Radio Frequency

1. Both sides know the used elliptic curve.
2. The smart phone generates one key pair consisting of public and private key.
3. The smart phone sends public key to the NFC-I over NFC.
4. The NFC-I also generates two key pairs. One key pair holds the own keys and the other key pair holds a random private key and the public key of the smart phone.
5. The NFC-I sends the public key to the smart phone over NFC.
6. The smart phone generates another key pair and copy the public key of the NFC-I in this key pair.
7. Both sides execute Diffie-Hellman algorithm, as input they get the public key of the communication partner and the own private key on the before agreed curve parameters.
8. At the end both sides have the same shared secret.

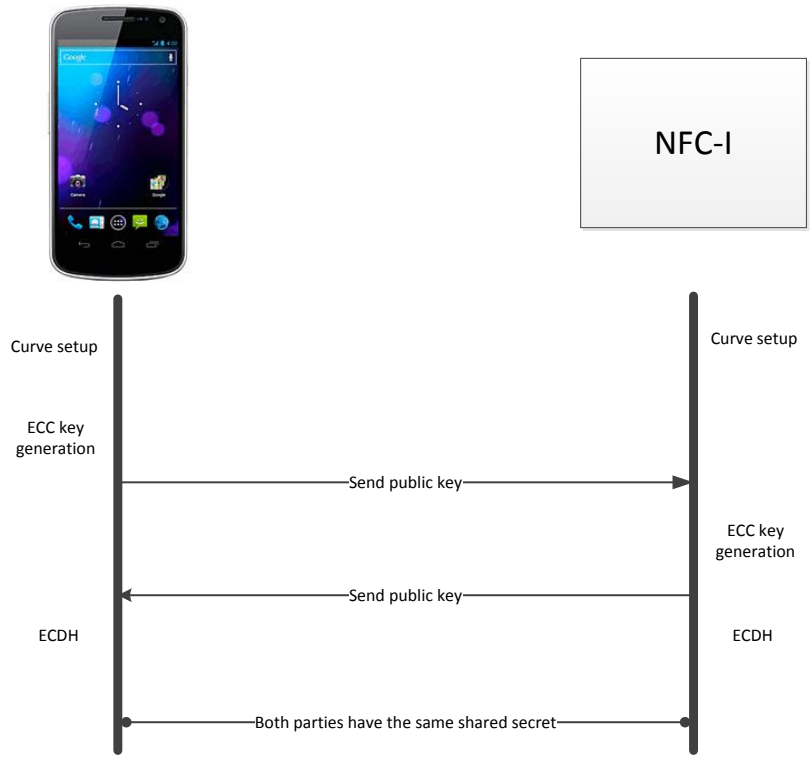


Figure 3.6: Communication flow of ECDH between the smart phone and the NFC-I.

3.2.2 Cryptography - ECDSA

The introduction of the digital certificates shall bring protection against manipulation of the data during the transmission to the communication partner and refuse the access to not authorized persons.
Like an ECC key pair, also a digital certificate consists of a private key to sign and a public key to verify the to signed message. In this way it is possible to verify if the message is really signed from the desired communication partner. To guarantee that the transmitted data was not modified, the introduction of hash-functions is needed. Hash-functions guarantee that every message produce a unique hash value, which will change by flipping one bit of the message and can easily be detected.
Figure 3.7 shows the flow of how a ECDSA with the NFC-I is done. The following steps describe the flow.

1. Both sides know the used elliptic curve.
2. The smart phone generates one key pair consisting of public and private key.
3. The smart phone hash the public key with a hash algorithm to protect it against manipulation.
4. The smart phone signs the hash value with the private key of this own digital certificate.
5. The smart phone sends public key and signed hash value to the NFC-I over NFC.
6. The NFC-I verify the signed hash value with the public key of the digital certificate from the smart phone finding out if the message is really from the desired communication partner and if it was manipulated during the transmission.
7. If the result of the verification is valid the normal procedure can continue, otherwise an unauthorized partner try to access the system.
8. The NFC-I sends the public key of the smart phone to the target device.
9. The NFC-I waits to receive the public key from the target device. Once received, the public key of the target device is send to the smart phone without any change over NFC.
10. Now the smart phone and the target device have all the needed information to calculate the shared secret.
11. At the end the smart phone and the target device have the same shared secret and know that the message was not manipulated and it was really an authorized partner.

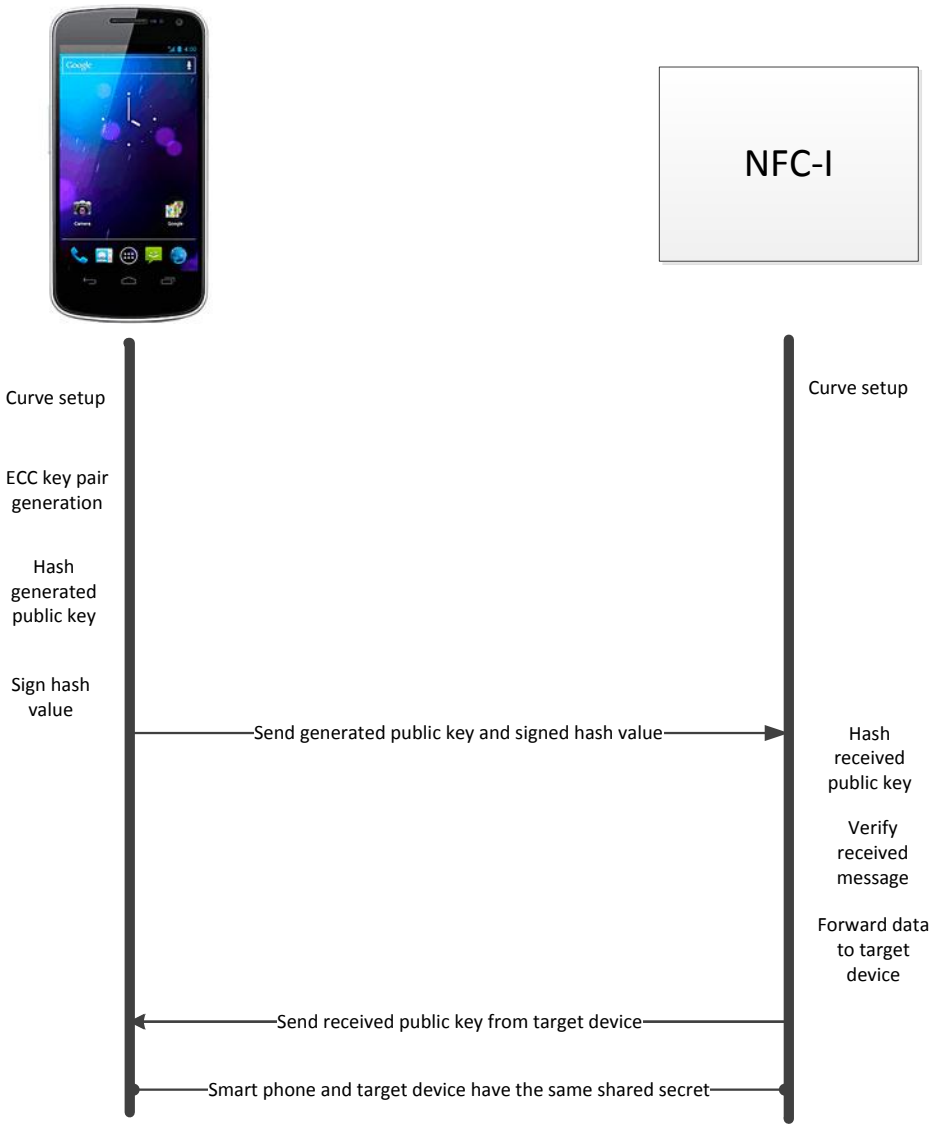


Figure 3.7: Communication flow of ECDSA between the smart phone and the NFC-I.

3.3 NFC-I

The NFC-I is used as the bridge between the smart phone and target device. It provides all kind of cryptographic methods from asymmetric to symmetric and give the possibility to communicate contactless with the smart phone and contact base to the target device as well.
The only missing part which is needed for the ECDSA is the same hash algorithm as used in the smart phone, to generate the same hash output for the message to verify. This

function has to be provided by the programmer.

Two different variants will be presented for the public key exchange. They create the appropriate environment for the operation of the NFC-I.

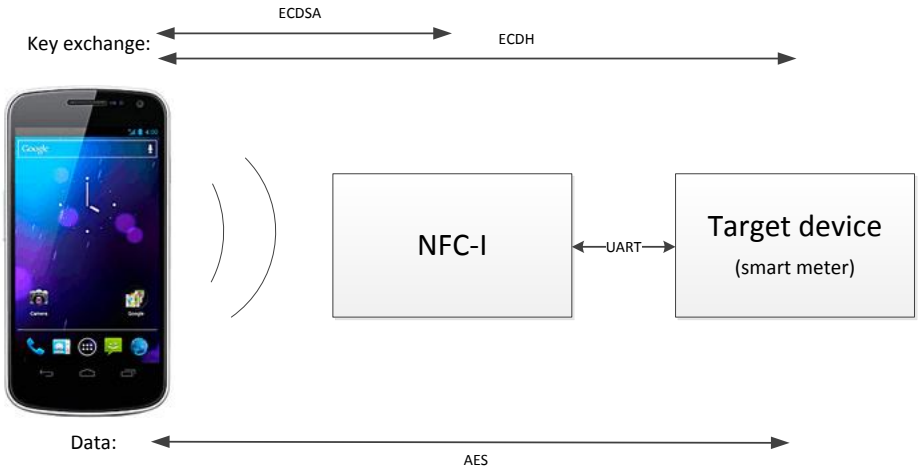


Figure 3.8: Overview on NFC-I, variant 1.

1. *Variant 1:* The smart phone sends its public key and its signed public key over NFC to the NFC-I. Once the NFC-I receives this data, it can start with the verification of the digital signature of the signed message. If the verification is valid, than the NFC-I forwards the public key to the target device. Otherwise the smart phone has not a valid digital certificate and the access to the target device cannot be given. Once the target device has generated its own public key and transfered to the NFC-I. The NFC-I just forwards the data over NFC to the smart phone (Figure 3.8).

2. *Variant 2:* The smart phone sends its public key over NFC to the NFC-I. The NFC-I sends this information directly to the target device. In this case the NFC-I is just a gateway between the smart phone and the target device that transfers the received NFC data over the serial port to the target device and vice versa (Figure 3.9).

Once the smart phone and the target device have the same shared secret, than the transmission of the data can begin. For the transmission of the normal data, the NFC-I acts in both variant in the same way, it forwards the data in both directions.

3.3.1 Program flow on the NFC-I

Figure 3.10 shows the simple program flow of variant 2. In this case the NFC-I acts like a gateway and just forwards the received data over NFC to the serial port. Afterwards the NFC-I wait to receive data over the serial port to send it back over NFC to the smart phone.

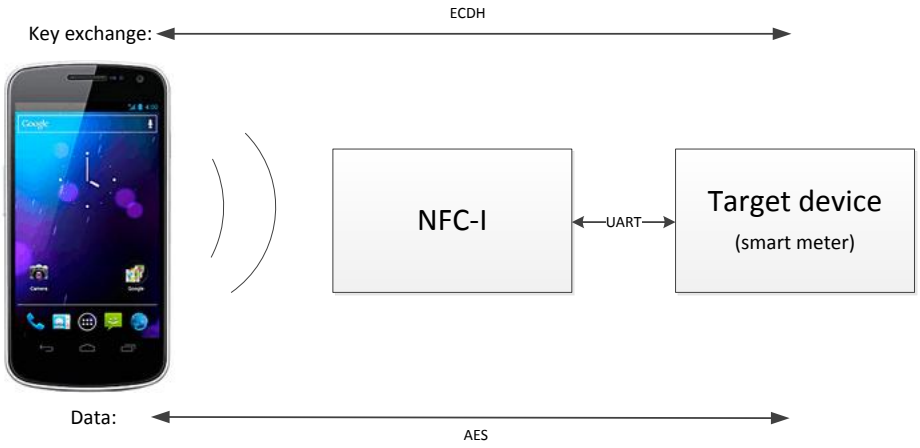


Figure 3.9: Overview on NFC-I, variant 2.

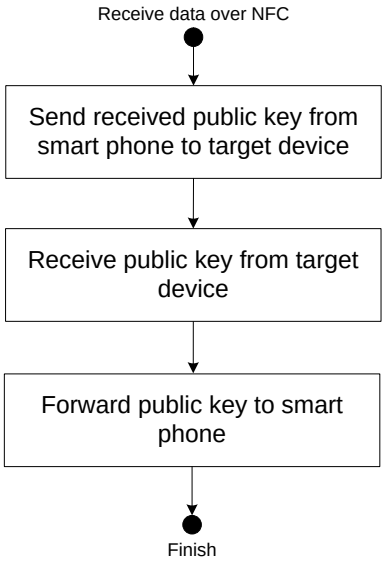


Figure 3.10: Program flow on NFC-I, variant 2.

Figure 3.11 shows the program flow of variant 1 where the NFC-I plays an important role for the verification of authorized access to the target device. Once the NFC-I receives the signed public key and the public key from smart phone, it can start to load the needed curve parameter for the verification of the signature. The second step is to hash the received public key with the same hash algorithm that the smart phone has used. The next step is to verify the signature with the hash public key and the signed public key. If the signature has not been signed from a known certificate, than the NFC-I returns an error

over NFC to the smart phone and the access to the target device is refused. But if the signature is valid, than the NFC-I forwards the public key to the target device and wait until the target device return its public key. Once the public key of the target device has been sent to the NFC-I. The NFC-I send this public key without any modification direct to the smart phone over NFC. Now, the smart phone and the target device has all the needed information to calculate the shared secret.

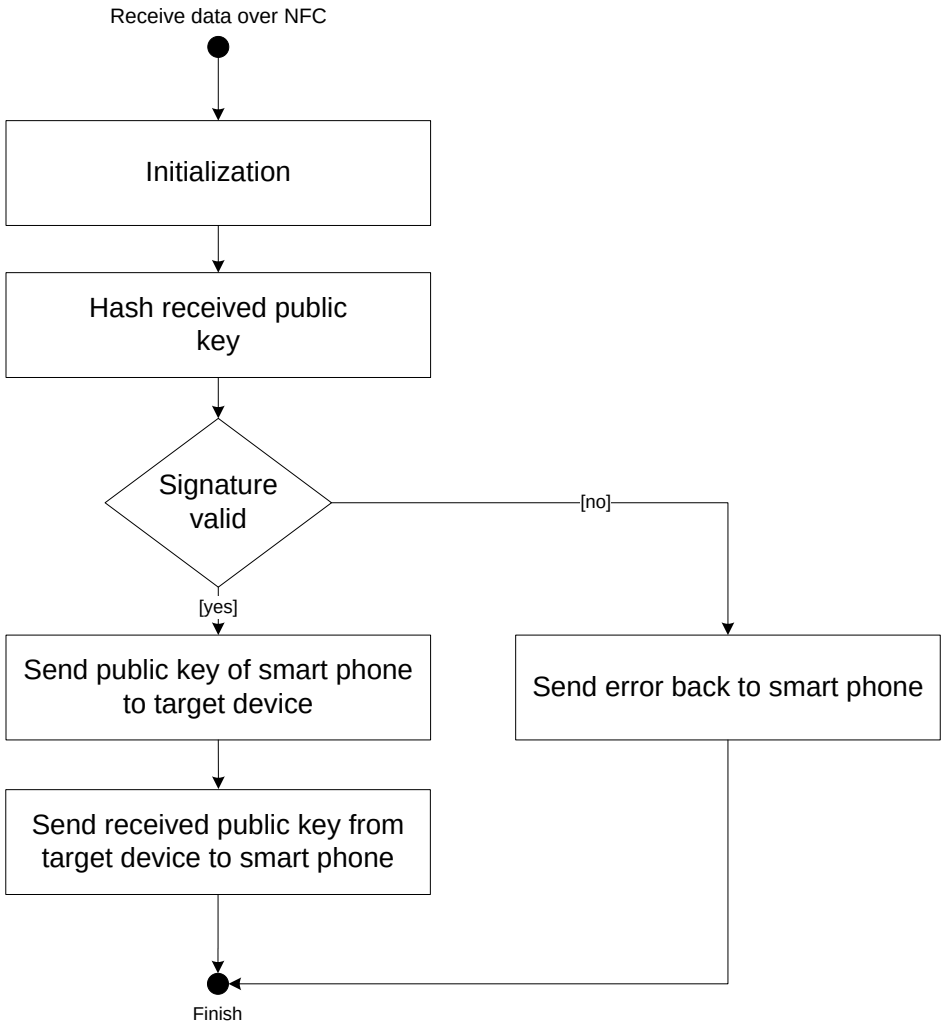


Figure 3.11: Program flow on NFC-I, variant 1.

Once the shared secret calculation is done and both sides have the same key for the AES encryption, the NFC-I just works as a gateway and forward the encrypted data in both directions, like in variant 2.

3.3.2 Hash algorithm on the NFC-I

As pointed out before for the ECDSA support a hash algorithm is needed. Due to the fact that no hash support is provided by the NFC-I, the programmer has to provide it by his own. Therefore, $SHA-1$, as most suitable algorithm has been chosen.

3.4 Communication channel between NFC-I and target device

The communication flow is very similar to the one between smart phone and NFC-I. The NFC-I forward the APDU messeger to the target device over a one line serial communication. This serial communication is used to transfer APDU messages to and from the target device. The transmitted data is the same which is coming from the smart phone or from the target device. Only during the ECDSA process the target device will not get the signed public key of the smart phone because it is not needed. In all the other cases the same data as over NFC is transmitted.

Figure 3.12 shows how one data frame is structured. The first bit of every frame is the start bit, this bit signal the receiver, that now data will follow. The next 8 bits are data bits. This is the information which both communication partner process afterwards. The next bit is the parity bit, with the help of this bit communication errors can be detected, but only when one bit fails. If two bit fails this error detection is not working any more. At the end two stop bits are transmitted, this bits signal the receiver, that now one frame is finished. For the transmission of more then one byte, a desired amount of this frame can be concatenated.

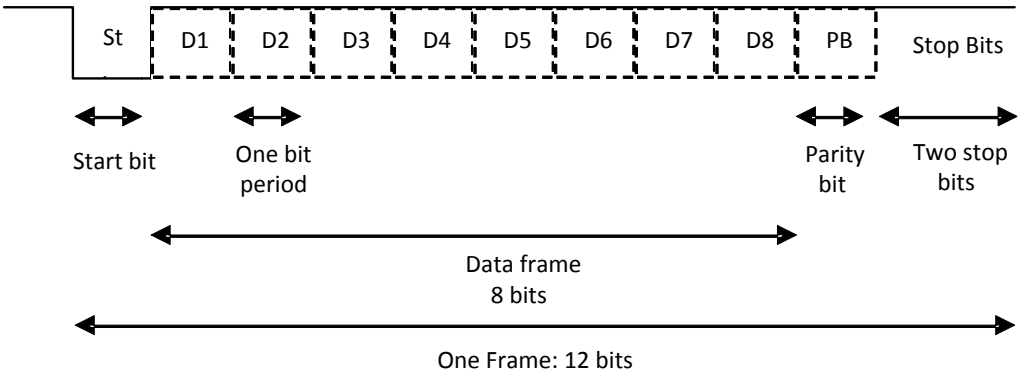


Figure 3.12: Structure of one frame sent over the one line serial port.

3.5 Target device

The smart phone will exchange data with the target device. Therefore, the target device needs some components. For the communication to the NFC-I the target device needs a serial port module accessible over GPIO, which supports ISO 7816 because that is the communication standard which the NFC-I uses. For the asymmetric and symmetric cryptography to the smart phone either a hardware module or a software cryptographic library is needed. The smart phone want read out sensor data, therefore, a sensor module is also needed. At the end, all this components are controlled over a CPU[4].

3.5.1 System overview

The system on the target device side consists of different parts:

- AES core: encrypt and decrypt the data which is received or send to the smart phone.
- Sensor: Collect data, to be processed on the CPU.
- GPIO: Communication channel to the NFC-I.
- CPU: Processor which controls all the other components and executes the software.

Figure 3.13 shows the system overview of the target device. The main component is the AMBA bus which connects all important components and peripherals with the CPU. On this bus also a hardware based AES core is connected. With the help of this AES core, encryption and decryption of data to the smart phone is possible.
The communication to the NFC-I is done over the GPIO. This GPIO, with the help of a serial port hardware module, can establish a communication to decrypt and encrypt the data transfer.
Data which is coming out of the sensor is transfered to the AES core to be encrypted and afterwards sent over the GPIO.
The last connected component on the AMBA bus is the CPU, it controls all the flow of the other components by setting certain bits.

3.5.2 AMBA bus enryption

Figure 3.13 shows how the encryption of the AMBA bus is done. The encryption of the bus is an essential part of the communication. Otherwise parts of the whole communication path would be again unprotected.
The sensor has to encrypt the data before putting them on the AMBA bus and the AES core decrypt it again before processing the AES encryption. Because of the high capacities of the AMBA bus it is very important to crypt data before transferring sensor data to the AES core. These high capacities create on every switch a power consumption which can be detected by side channel attacks and afterwards processed to information. From this processed information, valid data can be extracted. Other connection paths create a power consumption during the switching as well, but it is lower than the bus capacities. This is one of the main reasons why the data on the bus has to be encrypted.

[4]Central processing unit

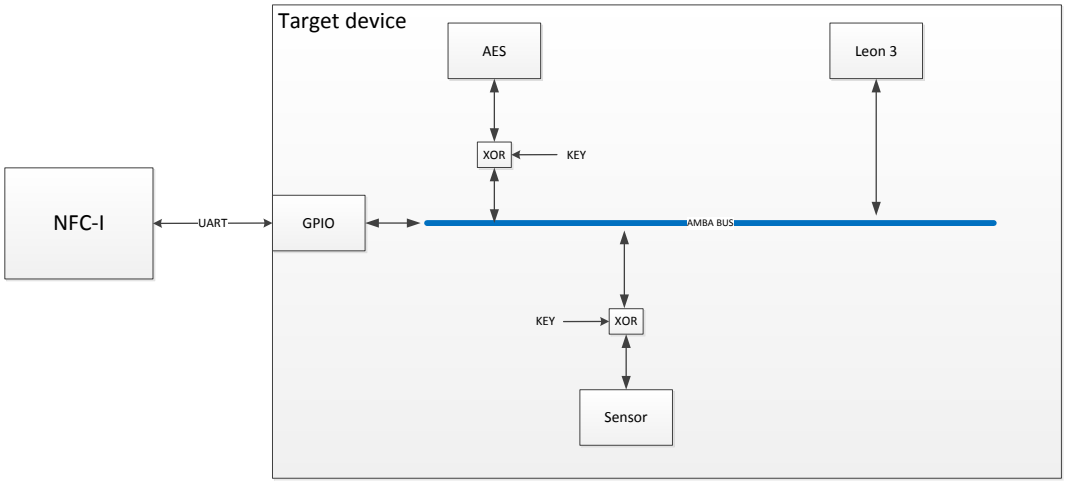


Figure 3.13: Target device, system overview.

3.5.3 Elliptical curve cryptography

Due to the fact that, currently, no hardware module of a ECC is available on opencores[5], the decision was to use a software module. Therefore, a cryptographic library is needed. This library should be designed for low power micro controller that not to much of energy is wasted for the shared secret calculation and key pair generation.

3.5.4 Serial communication to NFC-I

Figure 3.12 shows the signal flow of the serial communication. The receive/transmit line is always on high level. If the NFC-I or the target device want to send something, the line goes from high level to low. As one can see, the first bit is the start bit, the second until the ninth bit are the eight data bits followed by a parity bit which is used to control if there was a communication error and at the end two stop bits. In total for every frame 12 bits are transmitted at a speed of about 40000 *bps* which means that one bit is about $25\mu s$ long available on the line. In this short period of time the sent bit has to be detected by the communication partner, otherwise the communication fails. The used communication standard is ISO 7816.

3.5.5 Software program flow on the target device

Figure 3.14 shows the software program flow executed on the CPU. The program is waiting until new data is received from the NFC-I and starts to work depending on the existence of a shared secret:

- *Shared secret exists:* If a shared secret already exists, it is used as key for the AES encryption and decryption. To be able to decrypt the incoming data, only two steps

[5]http://opencores.org/

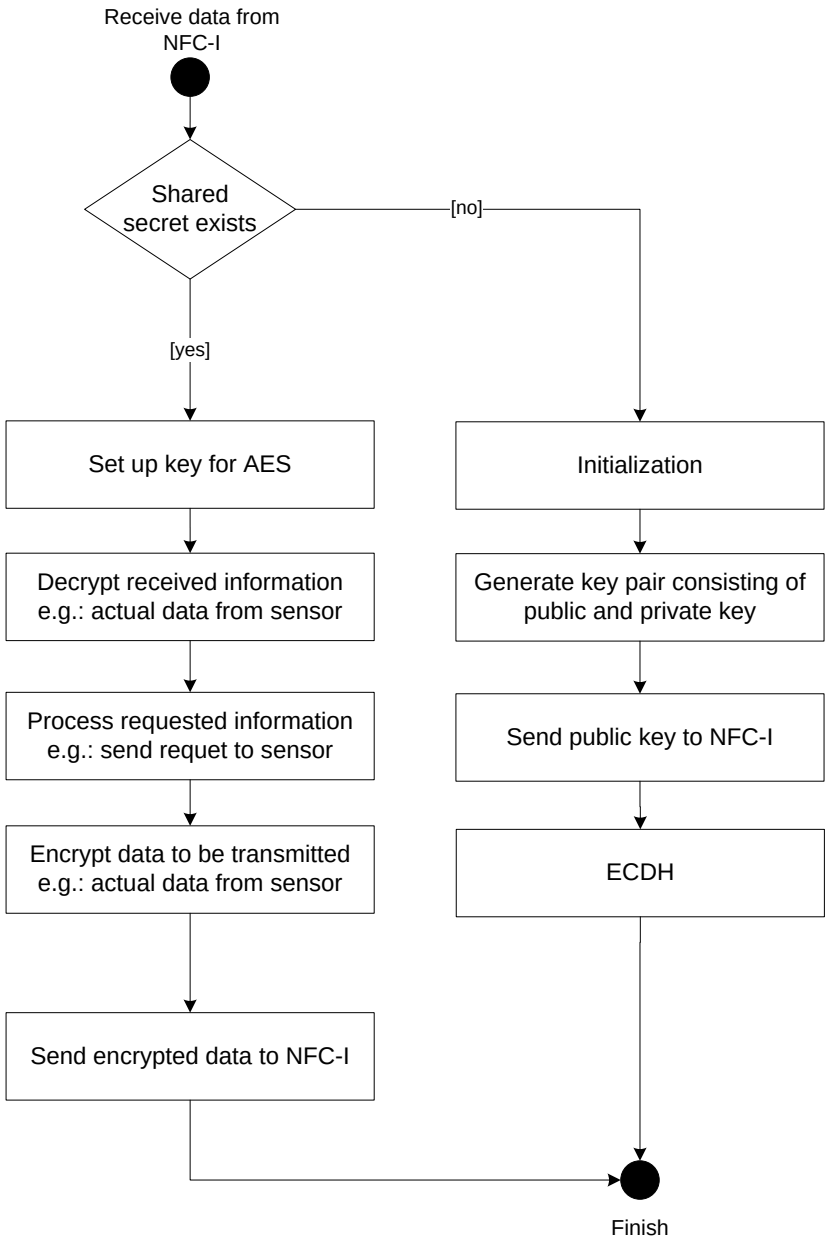


Figure 3.14: Software flow on the target device.

are needed. The first one is to setup the key for AES and afterwards to configure the AES for decryption. If the decrypted data is a request to get the actual value of the sensor, the CPU reads out the correct register and encrypt the data before sending them back to the NFC-I.

- *Shared secret does not exist:* If no shared key exists, the first step to do is to initialize the curves which are needed for the ECC. As second step a key pair has to be generated, afterwards the public key of this key pair can be send to the NFC-I. As third and last step the shared secret calculation can be done.

3.5.6 Class diagram

Figure 3.15 shows the class diagram of the target device. The only possibility of interacting with this program is to start it or stop it. The user application once started can access different libraries, e.g., the cryptographic library for ECC, the AES library for the AES encryption/decryption, the uart communication library for the communication with the NFC-I and access direct the sensor module to get sensor data. With the access of the libraries the whole program flow can be guaranteed.

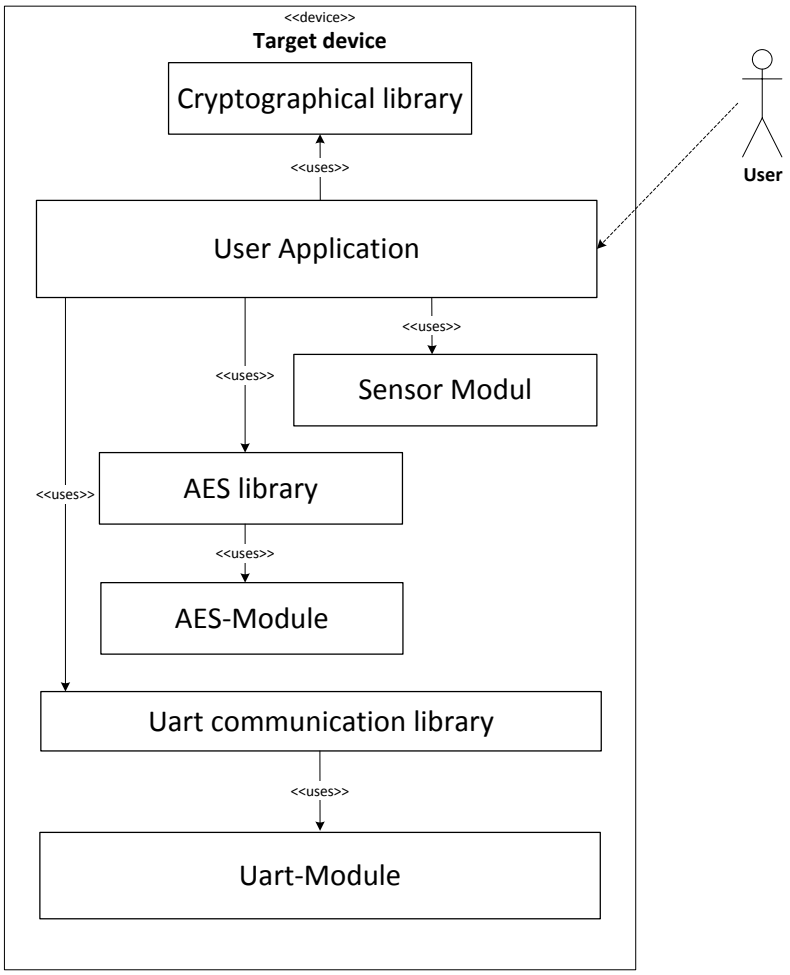


Figure 3.15: Class diagram, target device side.

Chapter 4

Implementation

This chapter details the implementation of the system defined in Chapter 3, the communication between the different components and each of the three communication partners. Furthermore, the problems which came up during the implementation and the reason why this kind of implementation was chosen is explained.
The Implementation was done on the smart phone, the NFC-I and the FPGA.

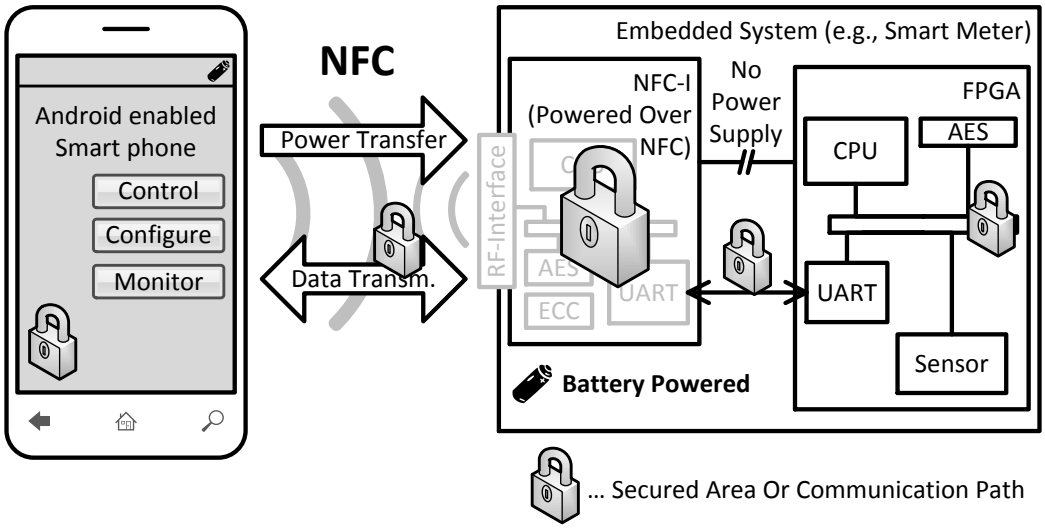


Figure 4.1: Implementation overview, adapted from [MM].

On two components (smart phone and NFC-I) all the implementation is done only in software with the help of cryptographic libraries. However NFC-I uses some hardware modules to help with the processing of the data. Instead of the target device a SoC[1] on the FPGA is used. On the FPGA software and hardware have been implemented as well. The asymmetric cryptographic part and the flow control is done in software. The rest (serial communication, symmetric cryptography and the sensor) are implemented in hardware and connected with an AMBA bus to each other.

[1]System on Chip

For the implementation on the smart phone side a Samsung phone, in this case, the Samsung Nexus S, with android operative system is used. The reason why the Samsung phone was selected is that it is a widely used device, available at the university and provides NFC as well.

An important decision to take was which security level and which curves are more suitable to use on all three communication partners. For example, the smart phone has support for all curves in order to compute ECDH. However, for ECDSA only certain curves are supported (see Appendix A). On the NFC-I all curves are supported for ECDH and ECDSA so there are no restriction. At the end the software of the FPGA supports only certain curves over F_P (secp112r1, secp128r1, secp160r1, secp192r1, secp224r1, secp256r1, secp384r1 and secp521r1).
Finally, the choice of the curves to used, according to different security level was taken. Therefore, the curves $secp160r1$, $secp192r1$ and $secp224r1$ over F_P were selected.
The three digits long number in the curve name gives information about the security level. The higher the digit is, the more effort the attacker has to devote to attack the system. That implies also a higher energy consumption. For current systems, the recommended security level is higher than 160 bit, because since 2010 this security level is not supposed to be used [oST07]. However, through the develop of this master thesis, 160 bits are used to show a comparison of the energy consumption to the higher security levels. Key lengths like 192 or 224 bit are more state of the art issue.

Explanation of all the curves parameter used, for example, in appendix B:

- *p:* Is the description of the finite field F_P. The cryptography is done over it.
- *a and b:* Is the description of the curve $E : y^2 = x^3 + ax + b$ over F_P.
- *G, G_x, and G_y:* Is the description of the base point. Where G is the compressed form and G_x and G_y the uncompressed form.
- *n and h:* Is the description of the order and cofactors of the base point.

4.1 Smart phone side

The implementation on thesmart phone side is done on a Samsung Nexus S phone with NFC support. As operating system on the smart phone, is used android in version 4.1.2, which is one of the lastest available versions. The programming language on android is java. On smart phone side the whole implementation is done in pure software.
For the communication with the NFC-I on the smart phone a special chip is needed. This NFC chip allows sending and receiving of APDU messages as well.

4.1.1 Cryptography

This NFC chip does not support only sending and receiving of APDUs, it also brings cryptographic support for symmetrical and asymmetrical cryptography. This cryptographic

methods are provided from a hardware module direct on the chip. However, the access of this hardware module is very difficult and the communication to this module has to be protected with additionally cryptographic methods. Therefore, in this master thesis the complete cryptography on the smart phone side will be done in software with the help of a cryptographic library.

The computational power of actual smart phones is high enough for software implemented cryptographic methods. The library Bouncy Castle[Bou], which provides cryptographic API[2]s for JAVA and C#, is used. This library is a special cryptographic implementation available under android. On android, a custom version of Bouncy Castle is implemented out of the box, but to have the complete support, the library Spongy Castle has to be used. The name is different due to a conflict with the names, since there is already an implemented Bouncy Castle library on android, which does not provide the whole range of functions. Spongy Castle provides all the cryptographic functions that are needed for this master thesis like key generation for ECC, sign and verification of messages with digital certificates, key exchange methods like Diffie-Hellman and encrypt/decrypt for AES.
The reason for using this library is the good support, because it is widely used for cryptography.

We implemented the needed security enhancement according the design shown in Figure 3.3 and Figure 3.4. To be ready to transfer data from FPGA to the smart phone and vice versa, first a shared secret has to be accorded. The shared secret is accorded with a ECDH calculation. If additionally security is needed also ECDSA can be used in order to allow the access to the FPGA only to authorized persons. Once the shared secret is agreed, the data con be protected with AES.

4.1.1.1 ECDH

The design presented in Figure 3.2 includes also an adaption. Adaption 1 is needed to calculate the shared secret. This adaption runs in a own thread which creates two key pairs, each of them holding a private and public key. Two key pairs are needed, one for the smart phone and one for the communication partner e.g.: FPGA. The public key is extracted from the own key pair and sent in a APDU over NFC to the communication partner. In this case the APDU is: *0xC0, 0xF3, 0x00, 0x00, 0x30, 48 bytes of public key for a 192-bit curve.*
The generated key pair is changed with the security level. Table 4.1 shows the public key length of the different used security levels, which then are sent over NFC to the NFC-I. The selection of the security level is done over a public member variable. The variable than select the desired curve for the key generation.

[2]Application Programming Interface

Security level in bit	public key length in bytes
160	40
192	48
224	56

Table 4.1: Public key length sent over NFC depending on the security level.

A APDU message is structured in the following way:

1. *One byte device identifier (CLA):* With this byte the device, with which the communication should happen, get selected (in our case `C0`).
2. *One byte program identifier (INS):* With this byte different programs on the device can be executed. This option is used for choosing the two variants, with `F3` variant 2 NFC-I gets activated and with `F2` variant 1 starts to verify the digital signature sent from the smart phone.
3. *Two bytes variables for the program (P1 and P2):* This two bytes can be used for executing different program parts in a selected program. This options are not used in this master thesis.
4. *One byte for the length of the payload (Lc):* This byte defines how long the total following payload will be.
5. *Payload (Data Field):* Relevant data which are, depending on the chosen variant, forwarded to the FPGA or processed internally.

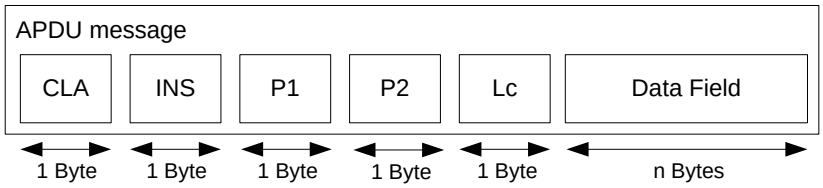


Figure 4.2: Structure of a APDU message.

4.1.1.2 ECDSA

The flow for the additionally digital certificates is quite similar to the ECDH, because the key pair generation is needed there. Once the key pair is generated, the signature procedure can start. Due to the fact that Spongy Castle only supports ECDSA for certain curves, in this master thesis only the $secp192r1$ curve is used (Appendix A shows the complete range of supported curves). This 192-bit curves produce as output two 24 byte long signatures (R and S). Both signatures and the public key which has been signed have to be sent to the NFC-I for verification.In this case the APDU is: *0xC0, 0xF2, 0x00, 0x00, 0x60, 48 bytes of public key for the 192-bit curve, 24 bytes for R and S part of the signature.*

For the distribution of the public key which is needed for the verification of the digital signature, different approaches can be used. The first one is to distribute the key over the internet. This method is not suitable in this project because the communication partner does not have an Internet access. Another possibility would be to sent the public key together with other information over NFC. For this project an easier approach is used, the public key, which is needed for the verification, is hard coded in the communication partner and vice versa.

Once the NFC-I has verified the origin of the signature, the NFC-I forwards the public key of the smart phone to the FPGA and send the public key of the FPGA to the smart phone. After both sides (smart phone and FPGA) have received the public key of the other, the shared secret calculation can be done (like described in Variant 1).

4.1.1.3 AES

The symmetric cryptography is used once a shared secret is calculated. The calculated shared secret is the basis for the AES key. The AES key needs a length of 128 bit, but the shared secret is longer than 16 bytes. This is the reason why only the first 16 byte of the shared secret are used, since both sides have to use the same bytes of the AES key. The rest of the shared secret is cut off and not used. Table 4.2 show the length of the shared secret in bytes.

Security level in bit	Shared secret length in bytes
160	20
192	24
224	28

Table 4.2: Shared secret length depending on the security level.

4.2 NFC-I side

The implementation of the NFC-I has been done according to the design described in Chapter 3.3. In the following subsection it will be described how we implemented the variant 1 and variant 2.

- Variant 1: The NFC-I plays an active role in the cryptographic process.
- Variant 2: The NFC-I plays the role of a transparent gateway, which only forwards the data.

4.2.1 The two different operation variants

As mentioned before, the NFC-I operates in two different variants. Depending on the APDUs which are sent from the smart phone, the operation variant on NFC-I side is selected and processed.

In variant 1 the NFC-I plays an active role for the cryptography. The smart phone sends over NFC its public key and its signature. The NFC-I need both public key and signature to verify the digital signature. The hashing of the public key is done with SHA-1. For details about the algorithm see Chapter 4.2.2.
The problems showed up during the implementation on the NFC-I side were, most of the time, connected with the memory management or the available size of RAM or ROM on the chip. Both of them were possible to solve with rewriting the program in a resource saving way.

In variant 2 the NFC-I receives the data over NFC and forwards it over the serial port to the FPGA. Due to the fact, that the NFC-I has only a general purpose (I/O) as defined in ISO 7816, the serial communication has to be done in software. The serial communication is implemented as a one wire communication. That means, that transmitting and receiving of data is done over the same wire. The serial communication is defined to be on high level $(3,3V)$ in idle mode, only if the NFC-I or the FPGA want to send something, the level on the wire goes low $(0V)$. On the NFC-I, when the data is sent to the FPGA a timer is started. Every time the timer underflows, a new bit can be transmitted until all the data is transferred. Depending on the bit, the wire is switched to high or low (see Figure 3.12 for more details about the bit order). After all, the data is transferred to the FPGA and the NFC-I waits until the data from thr FPGA is sent. Again in idle the wire is high and when the communication starts, it gets low and the NFC-I starts again the timer and the sampling of the bits. A second timer is started every time a byte is received in order to verify the end of the communication. Consequently, the second timer has to wait a longer time than the first timer, which has to sample the bits. Once all the received data from the FPGA is saved in a buffer, this buffer can be copied into the transfer buffer and the data is sent to the smart phone over NFC.

4.2.2 Secure Hash Algorithm - 1

Due to the fact, that the SHA-1 algorithm is public known and available on the internet. The decision was to port an existing solution to the NFC-I platform. The implementation was available[3] as c-code and it was very straight forward to port the software. The only problem that came up, is that the software was implemented for big endian and the NFC-I works with little endian. The exact program flow and mathematical explanation can be seen in the secure hash standard [oST].

4.3 FPGA side

On the FPGA (Xilinx Spartan3 XC3S1500-FG456) side a leon 3 [GAI] SoC from Aeroflex Gaisler is used. The leon 3 is a powerful open source processor which is based on a SPARC V8 architecture. On this CPU different programs are executed like a ECC software implementation and the flow control for the communication to the NFC-I.

[3] http://www.di-mgt.com.au/

4.3.1 Use case (smart meter)

The smart meter is one of the components on the FPGA, it emulates the behavior of a real smart meter. Figure 4.3 shows its graphical implementation in hardware. Both bus lines *DataIn* and *DataOut* (each of them 32 bit width) are connected with the AMBA bus, which connects all the components. In order to allow the access to the data from the CPU an additional register is placed on the AMBA bus. It stores the value for one cycle. This register can be easily accessed from the software. The incoming data from *DataIn* is the *DataOut* of the previous cycle. In this way the value gets never lost and can be increased every cycle. The first step is to decrypt the data coming from the bus because, as mentioned before, the encryption of the AMBA bus is very important to protect the system against attackers. The used cryptography on the AMBA bus is very easy, but efficient in terms of power consumption. It is done with an XOR operation. Before the data is sent over the AMBA bus, the counter value gets encrypted with an XOR and, before being increased, it will be decrypted again with an XOR operation. Both keys have to be the same, otherwise a wrong decryption would result. Once decrypted, the data gets inside the counter which is synthesized on the FPGA in form of an adder. This adder is clocked by the system clock and increases every cycle the value. The value is reseted when the reset line is high or the value pass a certain threshold (in the graphic called reset value). The sensor of the smart meter consists of the counter/adder part and the reset line, the rest of the implementation is used to ensure the security and to reset the counter after a certain threshold.

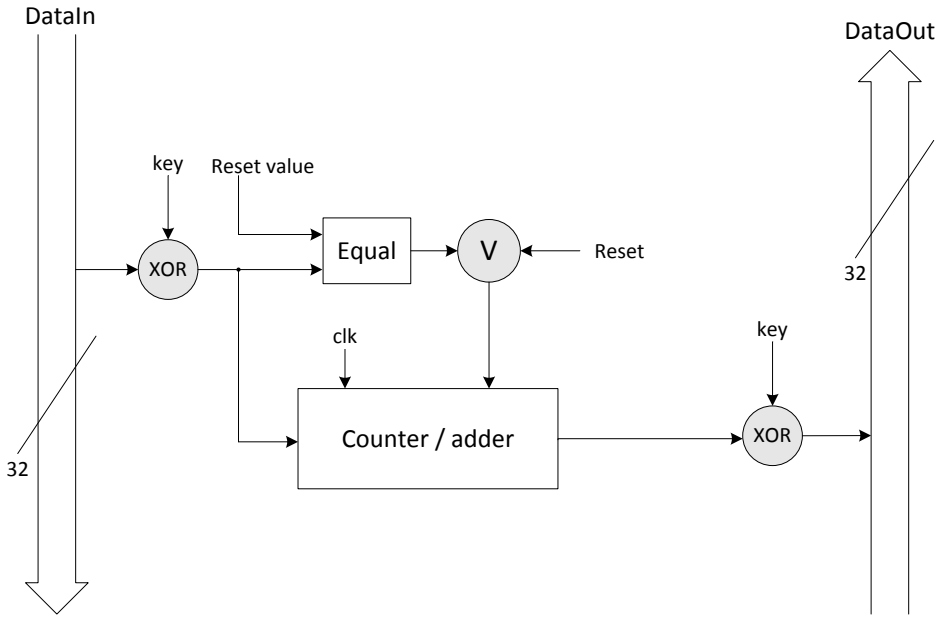


Figure 4.3: Graphical explanation of the implemented smart meter.

4.3.2 Serial communication from FPGA to NFC-I

Due to the fact that the NFC-I does not provide a serial port communication but only general purpose I/O's like it is defined in ISO 7816, the only way to implement the serial communication is to do it in software. The first idea was to implement also the serial port in software on the FPGA. It came up that the SoC was not really suitable for a timing relevant purpose. For a software implemented serial communication the timing is essential, because, like in a usual serial communication, the signal level is high when no data is transmitted. The transmission is started automatically when the signal level goes from high to low (Figure 3.12 shows the exact order of the bit). In order to achieve that the FPGA receives valid bits, the sampling of the bits has to be done in the middle of every bit, otherwise some bits would not be properly recognized and the receiver would receive wrong information. Some tests, which were carried out for this project, have proved that the FPGA needs about $20\mu s$ (which are about 700 cycles at a system clock of 30 MHz) when an interrupt is detected until the program is in the interrupt service routine. This system behavior makes the implementation of the serial communication in software almost impossible. That was the reason to implementing the serial communication in hardware.

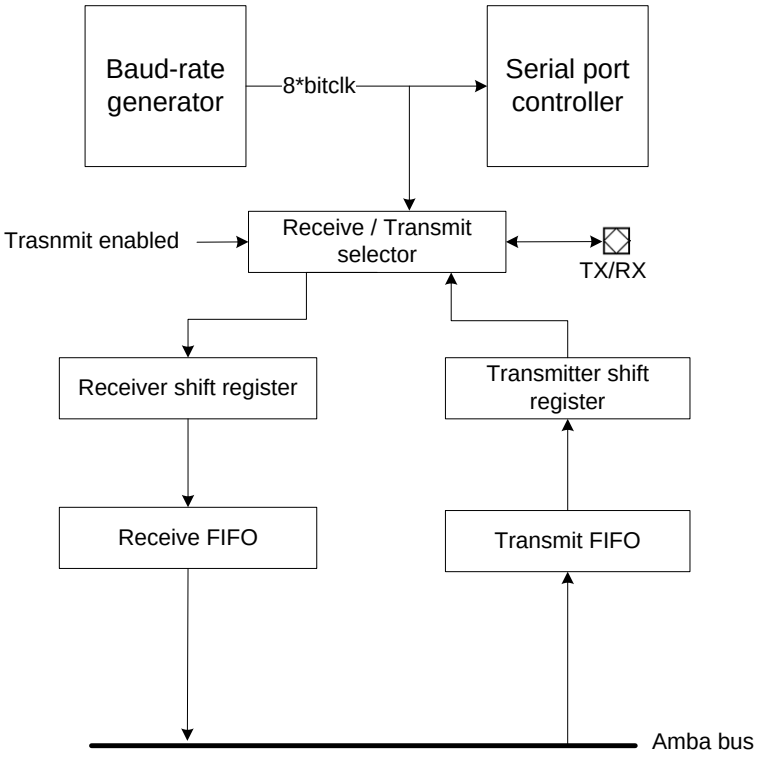


Figure 4.4: Hardware module for communication over serial port adapted from Gaisler leon 3 design.

Figure 4.4 shows the hardware model. The baud-rate generator generates a new clock, which is eight times slower than the system clock. With this clock the controller of the serial port is clocked. It is possible to break down this clock at the necessary communication speed. For receiving and transmitting an eight bit deep FIFO[4] buffer is used. These FIFOs are buffering the data to be transmitted and received. The control, read out and write data of the serial communication is done over registers (data, status, configuration and scaler), which are accessible over the AMBA bus.

Algorithm 6 shows the flow of how receiving data over a serial port is possible. The first step is to set the scaler for the internal clock generator. The generated clock has to match with the clock of the NFC-I otherwise a valid communication cannot be established. The second step is to configure the serial port. The *UART_CONFIG* activate the serial port in receiver mode with an even party bit. Only this two steps are needed to set up and configure the serial port. The next steps are needed to copy the received data. The TS bit is the Transmitter shift register empty bit. When this bit is logical high, data can be obtained from the data register and copied in a local buffer for guaranteeing the space in the FIFO for new incoming bytes. Once all the desired bytes are received, the while loop can be left.

Algorithm 6 Receiving data over serial communication channel

Summary: Receiving of data over serial port

```
1: define UART_SCALER 0x5C
2: define UART_CONFIG 0x21
3: counter = 0
4: receivedData = 0
5: maxData = 50
6: while 1 do
7:
8:    if  uartStatus = TS bit then
9:       receivedData(counter) = uartData
         counter = counter + 1
10:   end if
11:   if counter = maxData then
12:      break()
13:   end if
14: end while
```

Algorithm 7 shows the flow of how sending data over a serial port is possible. The first step is to set again the scaler to obtain the same clock as the NFC-I. The second step is to activate the serial port in send mode with an even party bit. To send some bytes, the FIFO has to be filled out. That can be done over the data register, which is used for receiving and sending data, by easily reading or writing from it. The next step is to check the status register. The TH (Transmitter FIFO half-full) bit informs about the free space that there is in the FIFO register. If this bit is logical high, new data can be copied to

[4]First In First Out

Algorithm 7 Sending data over serial communication channel

Summary: Sending of data over serial port

```
1: define UART_SCALER 0x5C
2: define UART_CONFIG 0x22
3: uartData = dataToSend[0]
4: uartData = dataToSend[1]
5: uartData = dataToSend[3]
6: uartData = dataToSend[4]
7: counter = 5
8: maxData = 50
9: while 1 do
10:
11:    if  uartStatus = TH bit then
12:       uartData = dataToSend[counter]
          counter = counter + 1
13:    end if
14:    if counter = maxData then
15:       break()
16:    end if
17: end while
```

the data register. Otherwise the transmitter has to wait. This procedure can be done as long as there is data to be sent. Once all the data is transmitted, the while loop can be left.

4.3.3 Cryptography

The cryptography in the FPGA is implemented in two different ways: a software implementation for the asymmetrical (ECC) part and a hardware implementation for the symmetrical (AES) cryptography. For the symmetric cryptography like AES there is a considerable amount of hardware modules available at opencore[5]. In this project the used AES core has been ported to this platform during an IT-project[6] by Daniel Kroisleitner and Johannes Samhaber. It has figured out that this AES core can encrypt and decrypt data very fast, for an encryption of 128 *bit* only 11 clock cycles are needed.

4.3.3.1 ECC

As mentioned before, all the asymmetric cryptography is implemented only in software due to the fact that no open source hardware module was available at this time. The basic of the software implementation is a cryptographic library[7], optimized for low power implementations on micro controller.

[5] www.opencores.org
[6] Emulation of a NFC Reader/Smart Card System on Prototyping Platforms
[7] TomLib

The flow of the ECC implementation is as following. As first step, the PRNG[8] has to be set up. As second step, the first key pair (public and private key) can be generated and saved. Than the program waits until the data is received over the serial port. Once the data (public key of smart phone) has been received and saved, the public key of the generated key pair can be exported. The key is exported to the ANSI[9] X9.63 standard and sent over a serial port to the smart phone. Once the key is sent, a second key pair can be generated. In this key pair the received public key from smart phone will be imported. Now, if the own private key and the public key of the smart phone are available, a Diffie–Hellman calculation can be done. The obtained shared secret has to be saved for a symmetric cryptography procedure afterwards.

4.3.3.2 AES

The symmetric cryptography is used, once a shared secret is agreed, to encrypt and decrypt the data stream to and from the smart phone. The smart phone sends a request to get information from the sensor and the FPGA answer them with the needed information.
The big advantage of using this AES hardware module is that it is very fast. It just needs 11 clock cycles at 30 MHz. That means that the encryption and decryption fulfills the requirements for this implementation. One disadvantage of this hardware module is the size. However, the FPGA has enough available place to store it (see Appendix C.1).

The flow of the interaction with the AES hardware module is as following. The precondition, which makes an encryption or decryption possible, is an agreed shared secret, which is done completely in software.
The first step is to wait until the data is received over the serial port. Once the data is received, the AES core can be initialized. The initialization step is needed to define the register, which allows the access to the core. The second step is to copy on this register the common key (shared secret agreed before). The next step is the decryption of the received data and its processing. If it is a request for the sensor data, the current value of the sensor has to be read out. Due to the fact that the read out data is cryptographically secured, also the AES core has to operate the same XOR operation than the sensor as first step. This XOR operation is done like in the sensor case, directly in hardware. The next step is to encrypt the sensor data and send it over a serial port to the smart phone.

[8] Pseudo Random Number Generator
[9] American National Standards Institute

Chapter 5

Results

This chapter shows the measurement results of the case study. The case study shows the energy consumption for reading out sensor data of a smart meter over NFC and display it on the smart phone over a secure channel.
In Figure 5.1 is shown the measurement setup. This chapter is split up in two main parts, one part is the energy comparison of different curves. Variants and possibilities are always measured with this measurement setup. The other part is the energy consumption on the FPGA, estimated with the power estimation unit and the results of the AMBA bus encryption.

This energy comparison should make easier to find a good balance between the security level to choose and the energy consumption. One has to consider that this comparisons are only valid for this test setup. If another cryptographic library or other curve parameters are used, the results can change and flip the decision for a better solution.
The results include two sources, the smart phone which is measured with the measurement setup and the FPGA where a power estimation has been done. A real measurement with the same setup on the FPGA would not make sense because other components on the evaluation board will falsify the results. This is the reason why on the FPGA a power estimation has been done.

The fact that the target device, which is powered over battery, should be able to run as long as possible without change of battery is an important issue of this master thesis. That can be guaranteed by the best choice between security level and energy consumption.

5.1 Measurement setup to measure the energy source of the smart phone

The Measurement of the power and energy consumption is done with a measurement suite. This suite has different components. The used system components are: the smart phone, the NFC-I, the FPGA, the measurement device (NI USB-6009) and a computer. The computer controls the measurement flow and post process the data.

Figure 5.2 shows the overview of the measurement suit. The measurement instrument samples over the two analog inputs, the current (I) and the voltage (V), of the smart phone. The current is measured over a potential drop with resistor (R = 1 Ω) and the

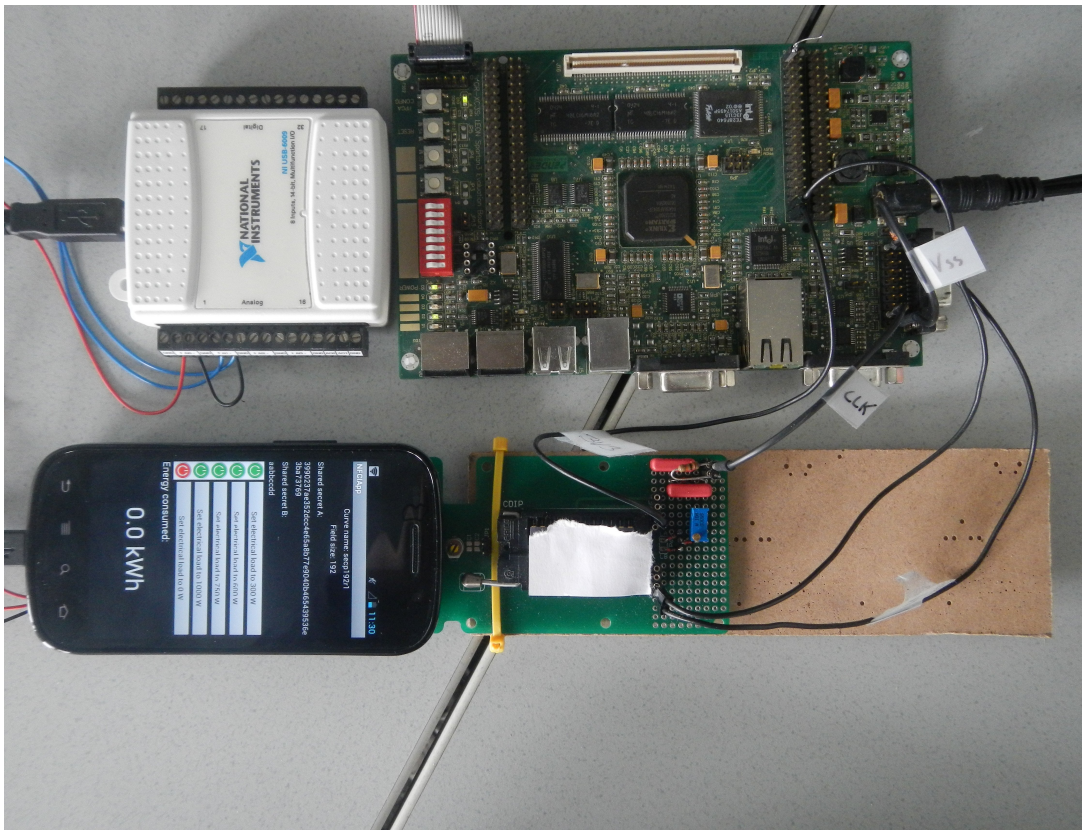


Figure 5.1: Measurement setup (general overview).

voltage is measured directly on the battery. With this two values it is possible to calculate the power consumption (Equation 5.1). To obtain the energy not only the power, but also the time (T) is needed (Equation 5.2). For every measurement this three values are required.

$$P = I \cdot V \tag{5.1}$$

$$E = P \cdot T \tag{5.2}$$

The voltage and current values are accessible directly over National Instruments APIs. This APIs are used from a C# program to capture all the values and save them in mat files which are needed afterwards for the post processing in Matlab.
There is a WIFI connection between the computer and the smart phone, this communication is needed for executing automatized test measurements. The computer sends over WIFI the command to execute certain functions on the smart phone. In this way it is possible to define test scenarios and the measurement suite measures them automatically.

The measurement, done with this suite, shows only the exact energy consumption of the smart phone and the NFC-I. The energy consumption of the FPGA is only indirect inside the energy consumption of the smart phone. The FPGA interferes only with its timing in the energy consumption of the smart phone. Differently to the NFC-I, it is powered by

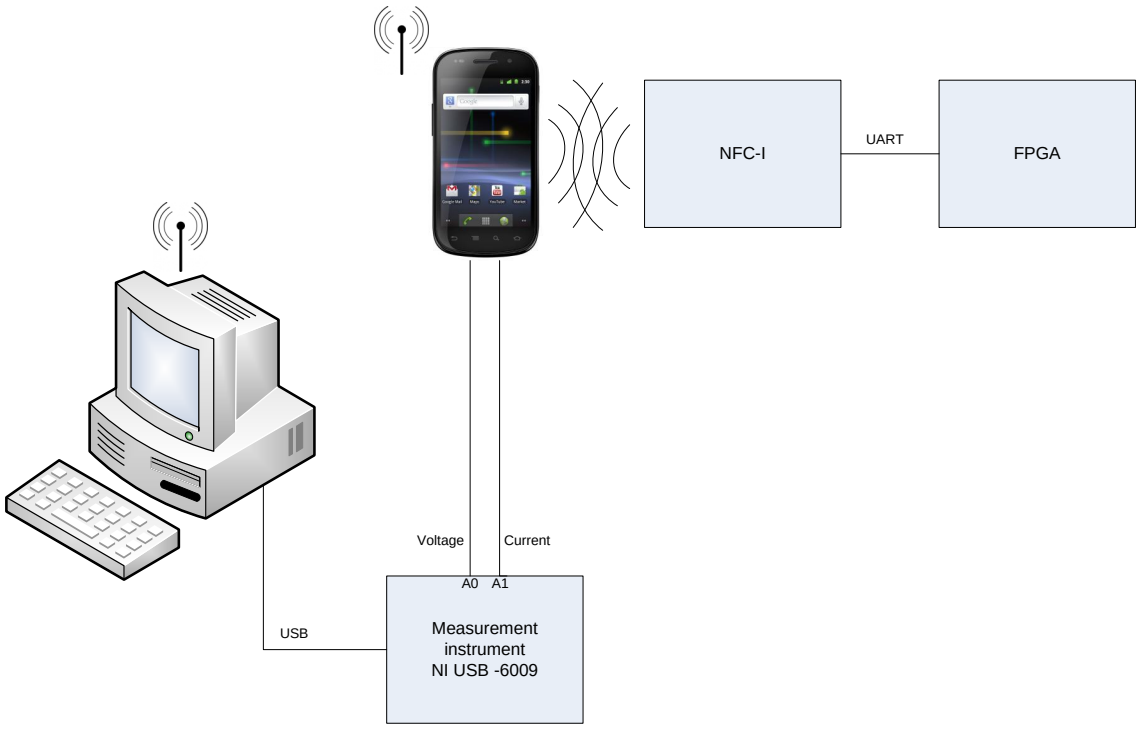


Figure 5.2: Measurement setup for the source of the smart phone.

the smart phone and included in the energy consumption of the smart phone. It would be possible to measure also the energy consumption of the FPGA with the same suite, but it would not be the real energy consumption, because other not used components, which consume energy, are on the board. This is the reason why only the smart phone and the NFC-I are measured. The energy consumption of the FPGA is estimated with a power estimation unit (for more details see 5.4).

5.2 Measurment of the communication for ECDH

The ECDH is needed in both variants and it is also the start point of every data transfer. That is the reason why it is important to know how much energy the ECDH needed and which factors influence this energy consumption. As described in the related work the longer the key is, the more energy is needed.

Figure 5.3 shows real measured values. The energy consumption grows while increasing key lengths. The three single plots show the needed power over time. In every plot the background energy (like WIFI, operation system processes, back light of the display etc.) is subtracted from the total energy to get an, as close as possible, estimation only about the real energy consumption of the communication and cryptography. An interpretation of the single periods in the plots is very difficult, because the background operations influence the comparability. This is the reason why the next comparisons are only done over the normalized energy. All normalized energy figure includes the background operations, so

deviations to the real value are given.

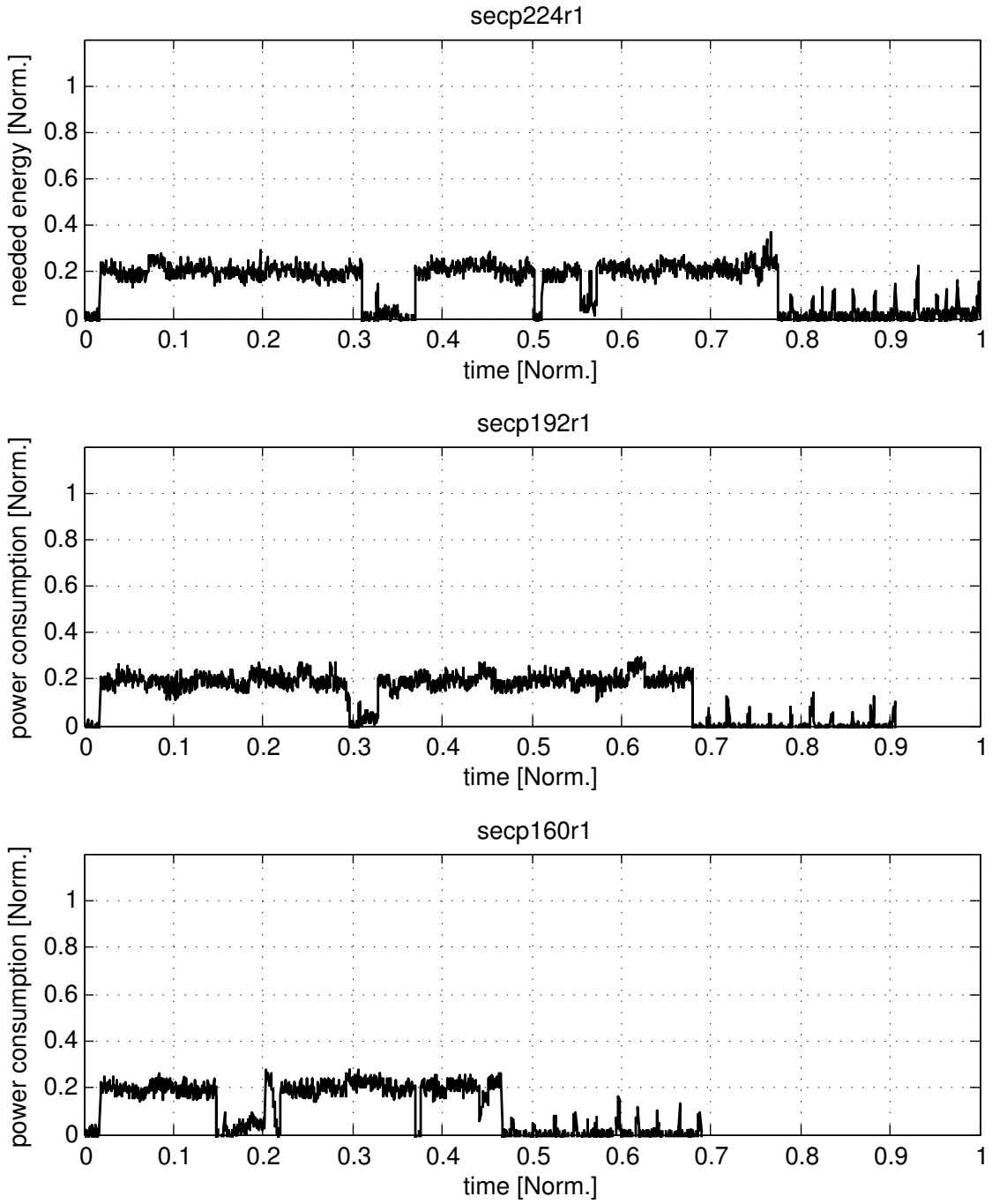


Figure 5.3: Power profile of different key lengths (Variant 2).

5.2.1 Comparison of different security leveles with the relative energy consumption (Variant 2)

As shown before in the Figure 5.4, the power over the time is not so significant. That is the reason why in this master thesis, the integrated energy over the time is used. With this representation it is easier to see the difference of the energy, if two different methods or curves are compared.

Figure 5.4 shows the difference of the used three elliptic curves. Although if it is not recommended any more to use 160-bit curves, in this master thesis is used to show the difference in terms of energy compared with curves of higher security level [oST07]. The 160-bit curve needed about 63 % of a 224-bit curve, so that the difference is about 27 %. This difference of energy shows that it makes sense, if cryptography is required, to use a higher and securer curve than an already not recommended one. From the 192-bit curve to the 224-bit curve the difference in terms of energy is about 12 %.

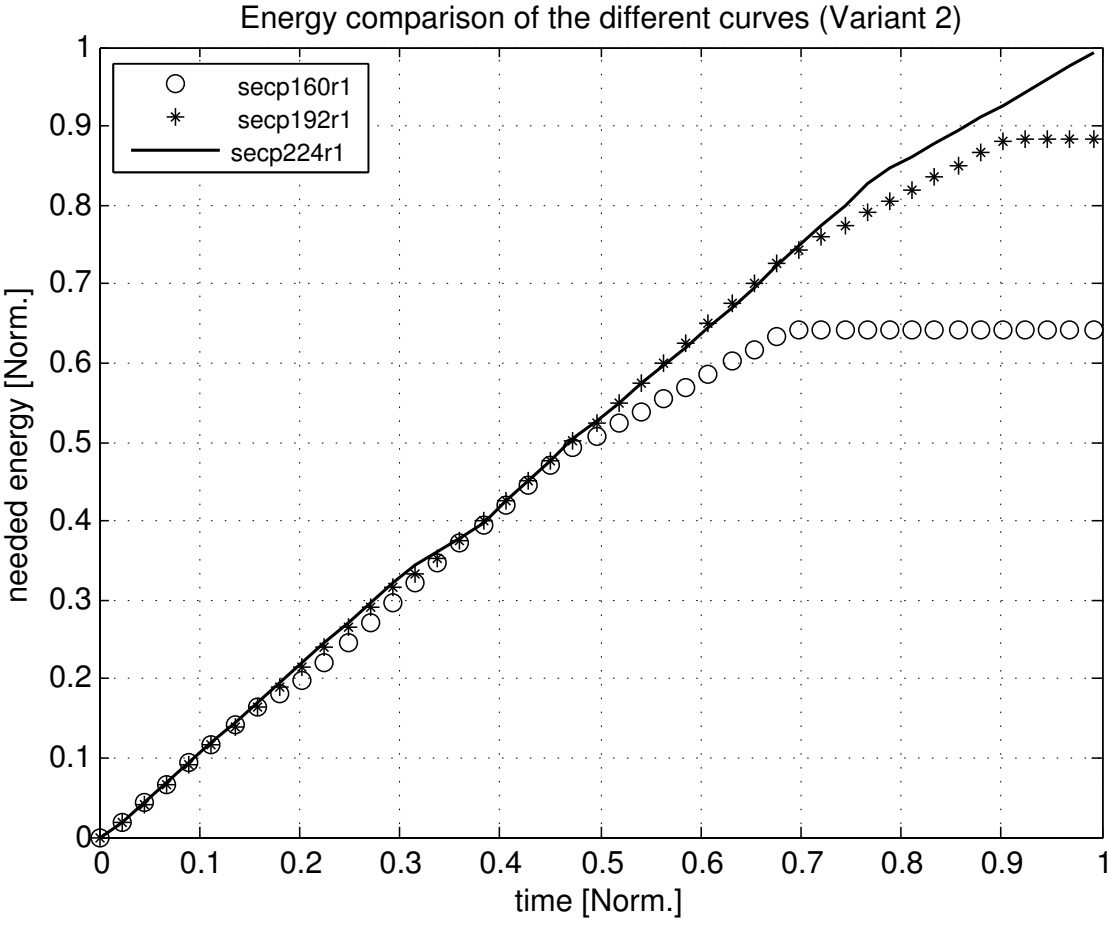


Figure 5.4: Energy comparison of the different key lengths (Variant 2).

5.2.2 Energy comparison of ECDH and AES (Variant 2)

Figure 5.5 shows the energy consumption of a ECDH shared secret calculation and a complete data exchange, protected with AES, from the smart phone to the FPGA. The flow for the ECDH is the same as in variant 2. The data exchange with AES protection is explained in the following steps:

1. The smart phone want to request sensor data from the FPGA, therefore, the request has to be encrypted and sent to the FPGA.
2. Once the FPGA has received the encrypted message, it starts to decrypt it and post process the result.
3. Get the actual data from the sensor.
4. Encrypt the data before sending it back to the smart phone.
5. To obtain the valid data, the smart phone has to decrypt the message again. Now the smart phone has received the desired information.

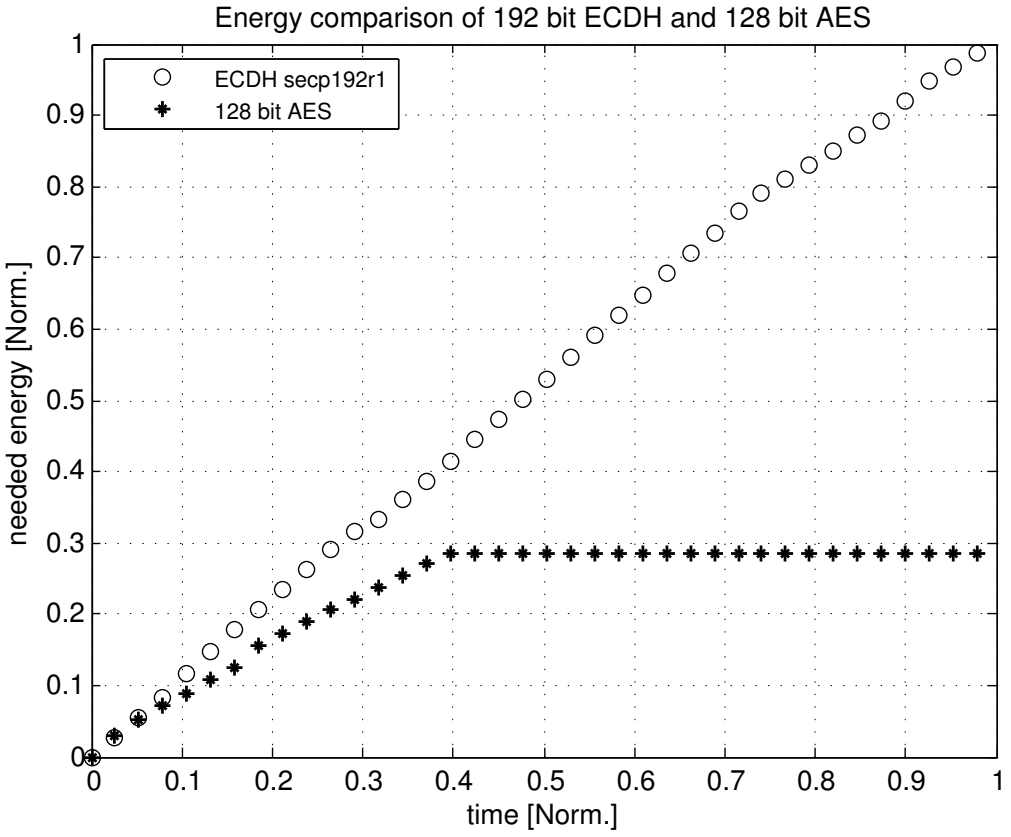


Figure 5.5: Energy comparison of ECDH and AES.

The amount of data which is send and also crypted is 128 Byte. It is possible to see that a complete sensor data request with AES protection need 30 % of a ECDH on a 192 bit

curve. Nevertheless, to obtain the shared secret which is needed for the AES encryption, a ECDH is needed.

5.2.3 Energy comparison of ECDH (Variant 2) and background

The background services are always running on a smart phone, also some of them during the display is switched off. This services check, if new updates are available, process operating system information etc. All this services need energy. To know how much energy they need, Figure 5.6 shows the energy comparison between a ECDH and the background services with activated WIFI, display and transfer power for NFC. One can see that more than the half of the energy which is needed for a ECDH is consumed from services, the back light of the display and the transfer power for NFC. Therefore, at the end, the energy which is needed for the ECDH is about 18 % more than leaving the phone switched on.

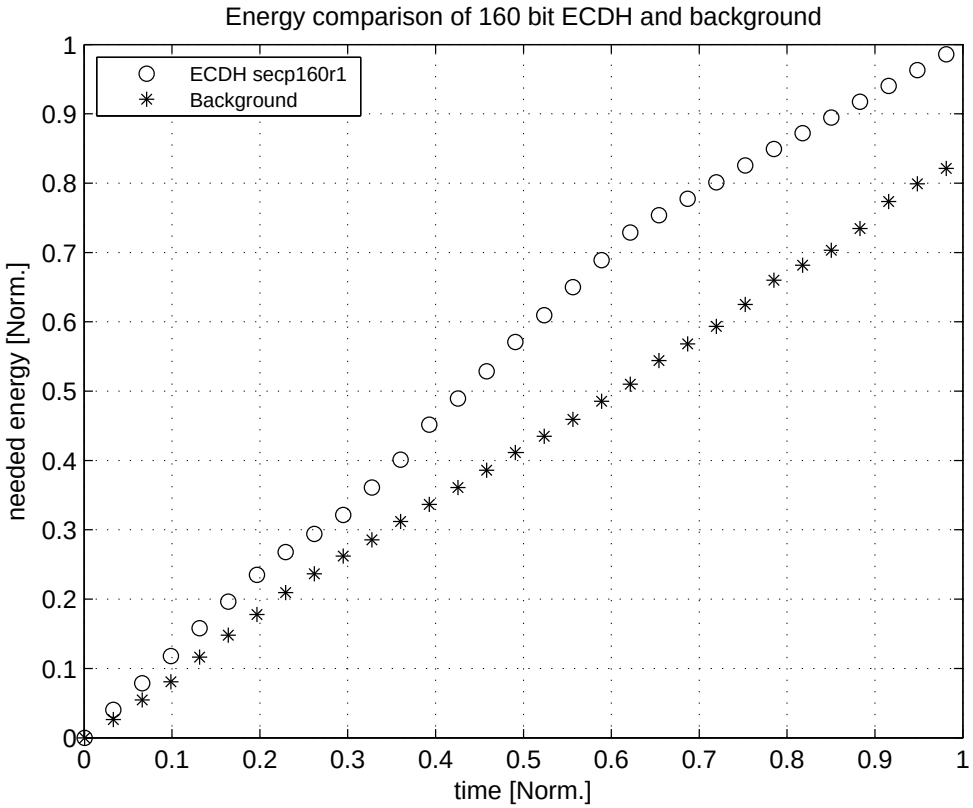


Figure 5.6: Energy comparison of ECDH and Background services.

5.3 Elliptic curve digital signature algorithm

The introduction of the digital certificates brings the benefit to increase the security, because only a person with a valid certificate can access the sensor data.

5.3.1 Evaluation of the trusted communication - ECDSA (Variant 1)

Figure 5.7 shows the energy consumption of ECDH on a 192-bit curve and ECDSA also on the same 192-bit curve. In this case ECDSA means, first sign the public key on the smart phone side, than check the signature on the NFC-I and if the signature is valid compute a share secret calculation between the smart phone and the FPGA.

One can see that the difference in terms of energy is about 20 %. That shows that the signature on the smart phone and the verification on the NFC-I needed 20 % more then a ECDH between smart phone and FPGA. The reason for that is, that the private key for signing and the public key for verification are both hard coded in the devices. Otherwise, the energy difference would be higher, because a extra key only for signing and verification would be needed. The big disadvantage of the hard coded method is that, if someone gets the private key of the smart phone which is used to sign the data, also unauthorized person can get access to the target device easily. In this project it is not a big issue, but in a commercial system it is recommended to generate each time a new public and private key for signing and verification.

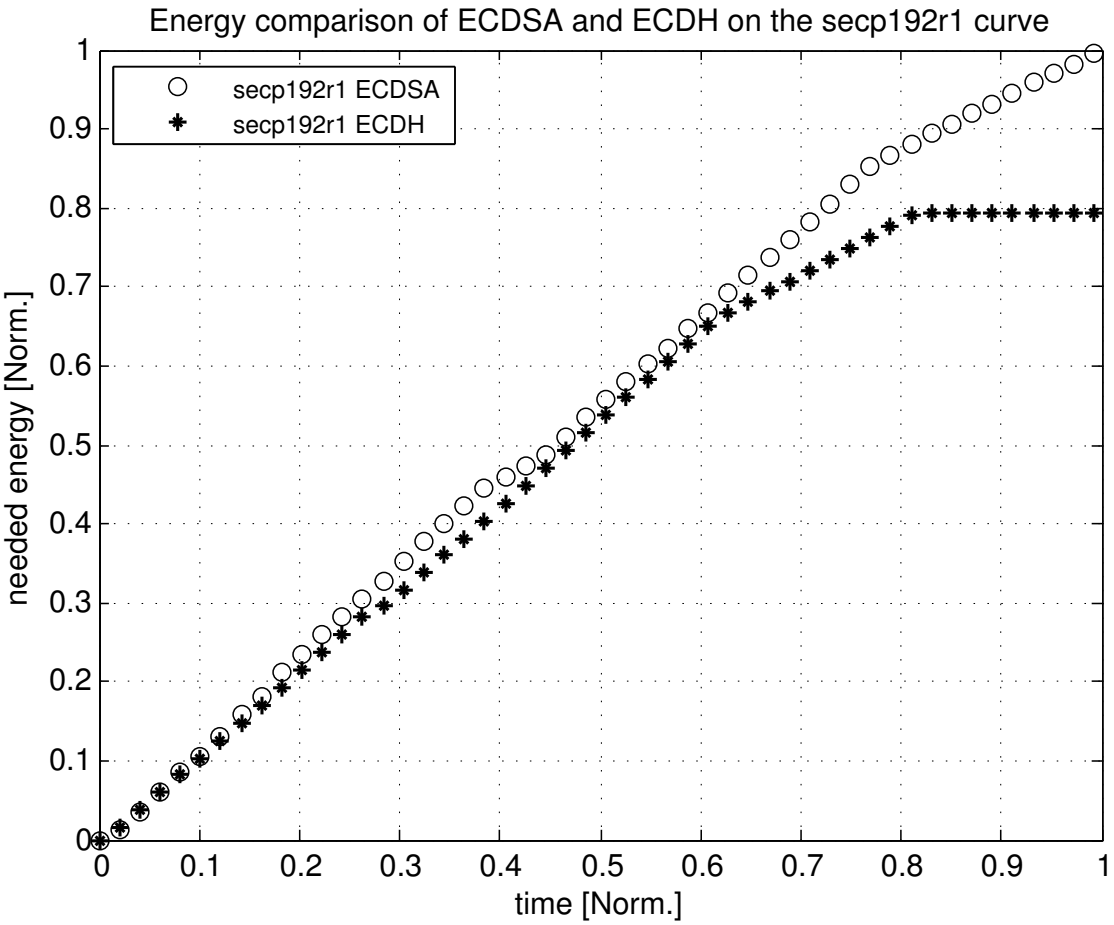


Figure 5.7: Energy comparison of ECDH and ECDSA.

5.4 Energy estimation of the FPGA

The measurements on the FPGA side are done over power estimation. Therefore, a modified version of the leon 3 is used. The modification includes a Power Estimation Unit (PEU) in the system, which uses the power characterization from [BGS+10]. This power characterization gives a cycle accurate estimation about the energy consumption of the executed software with an approximated error of about 2%. This power estimation can only be used for the ECC calculation because it is done exclusively in software. For the AES hardware core the power estimation can not be used to determine the energy consumption, because the power characterization of the AES hardware module is not included.

5.4.1 AMBA bus encryption

The AMBA bus encryption is one of the used strategies in this project to secure the complete communication path. Due to the fact that security also means more energy or more area consumption on a chip, an ideal combination has to be found. The system bus encryption is done with a very simple but efficient way by just adding an 32 *bit* wide *XOR*. The energy consumption of this *XOR* can be neglect because it is so low, but the most interesting fact here is that much more area is needed on the chip. If the chip area increase also the price of the production will increase.
On the FPGA this 32 *bit* wide *XOR* needs six slices and 13 LUT[1]'s on a Spartan 3 platform. To show that this invested area in security is really needed, table 5.1 shows the data on the AMBA bus without encryption. If someone get access to this data all the other cryptographic methods on the smart phone etc. are useless. Table 5.1 shows the data on the AMBA bus with activated encryption. One can see that there is no relation to the original data any more. If the attacker accidentally get this data, the attacker can do nothing with them, they are useless for him.
The extraction of the system bus data is done on simulation level with Modelsim.

Data on AMBA bus without cryptography	Data on AMBA bus with cryptography
00000000000000000000000000000010	00010010001101000101011001111001
00000000000000000000000000010011	00010010001101000101011001101010
00000000000000000000000000100000	00010010001101000101011001100111
00000000000000000000000010100001	00010010001101000101011011011000

Table 5.1: Some random picked data to proof the encryption of the AMBA bus.

5.4.2 Elliptic curve cryptography

As mentioned before, the power value for the ECC is obtained over the power estimation unit which gives a cycle accurate power estimation with an error of about 2 %. This power estimation unit can be used to give an early estimation about the energy consumption needed on the real system. With this estimation it is possible to see if the power supply over the contacless interface can be guaranteed. In this master thesis it is used to estimate

[1] Look-up table

the power consumption on the FPGA side.

Figure 5.8 shows the energy consumption of a ECDH shared secret calculation on FPGA side. In this case, two key pairs are generated and used for the shared secret calculation. One can see that the energy consumption is quite similar to the energy consumption in Figure 5.4. In both cases the energy grows with higher security level.

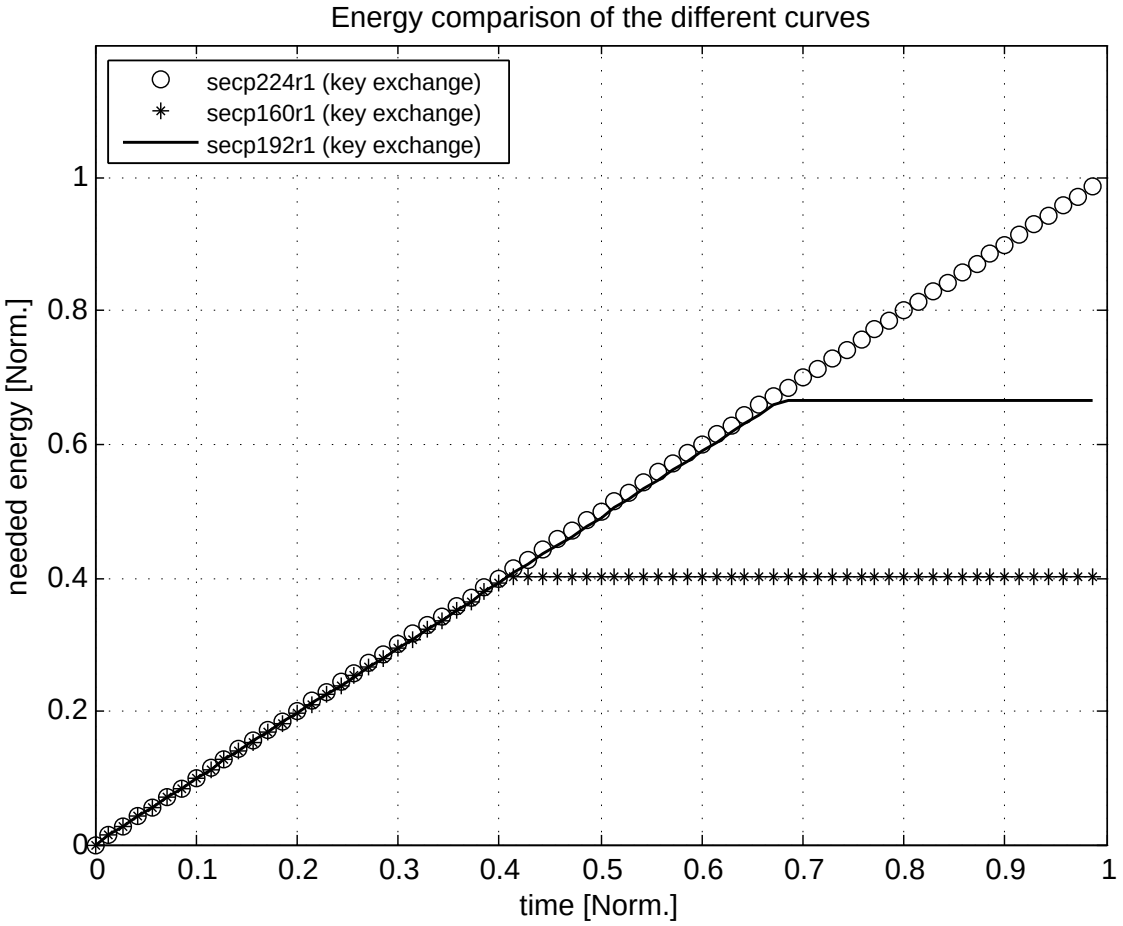


Figure 5.8: Energy comparison of different curves on FPGA side.

The cryptographic libraries used on the smart phone side and on the FPGA side are different. On the smart phone side a java library (Spongy Castle) is used and on the FPGA side a C library (TomLib) is used. These different libraries have also a different influence for the energy consumption.
Figure 5.9 shows the cryptographic library on the smart phone side, while increasing the key length the energy consumption changes the gradient. From 160 bit until 192 bit the gradient is higher than from 192 bit until 224 bit. This shows that there is no big difference in terms of energy between 192 bit and 224 bit curves.

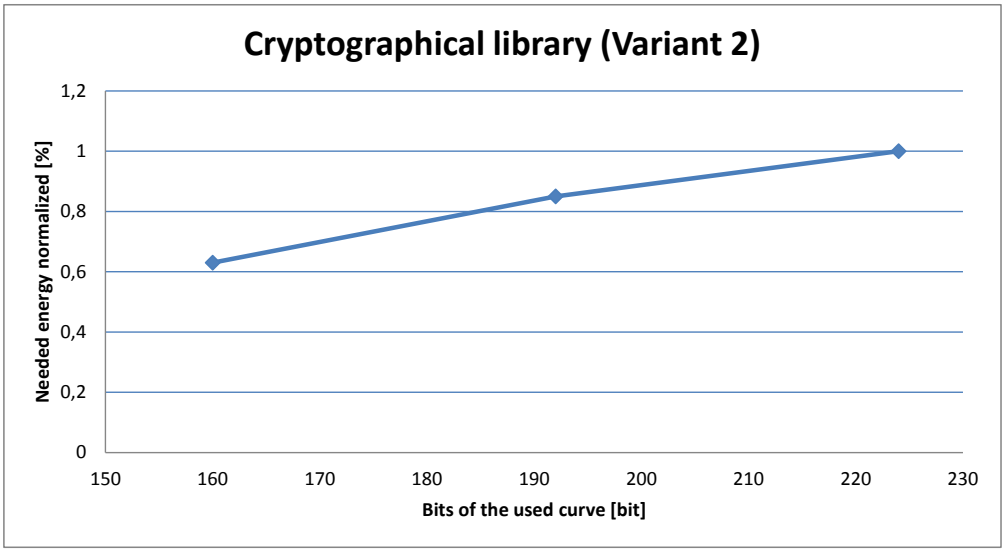


Figure 5.9: Energy evaluation of the cryptographic library (Smart phone side).

Figure 5.10 shows the cryptographic library on FPGA side. Here the gradient is constant from the 160 bit curve until the 224 bit curve. Differently to Figure 5.9 where the gradient is not constant and a higher security level costs more energy.

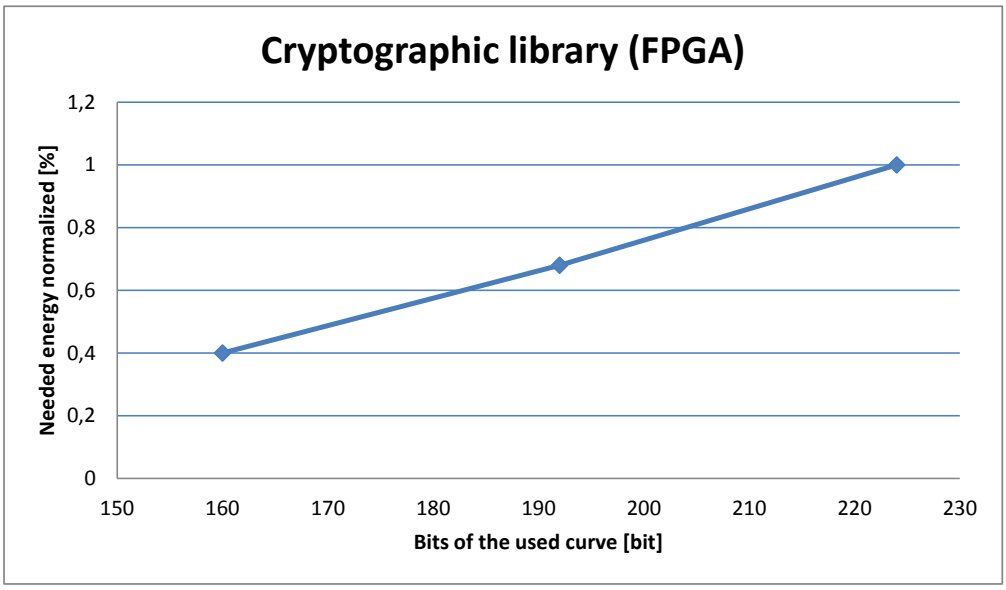


Figure 5.10: Energy evaluation of the cryptographic library (FPGA side).

Chapter 6

Conclusion

Nowadays smart phones provide enough computational power to process cryptography in software. This computational power in combination with NFC create new possibilities to use the smart phones. These new possibilities, like reading out sensor data (e.g. smart meter) or controlling actuators, bring also new chances for attackers to eavesdrop or attack target devices. These attack or eavesdrop chances should be as hard as possible in order to force the attacker to devote a lot of time to get useful information.

Therefore, one of the strategies used in this master thesis, is to protect the whole communication path. Most of the times, in current used system, parts of the system are not protected with cryptographic methods like, e.g. the system bus of the target device.

Another strategy is to use state of the art cryptography which is well know, mathematical proved and whose algorithms are open source. Only by using this state of the art cryptography a maximum of security can be guaranteed.

The results of this master thesis have shown that the evaluation of different strategies, is one of the possible ways to find the best constellation for the analyst system. Giving general suggestions for every system is not possible. Each use case has to be evaluated individually. All the evaluations done during this project are only valid for this precised use case.

For the Diffie-Hellman shared secret calculation in the variant 2, the difference between the 192-bit curve and the 224-bit curve is about 12 %. Therefore, it is recommended to use the 224-bit curve if a higher security level is needed. Although the difference in terms of energy consumption between the 160-bit curve and the 224-bit curve is about 35 %, since 2010 it is not recommended the use of 160-bit curves because of the short key length. If the energy consumption has really to be reduced to a minimum and the transferred data is not so critical in terms of privacy the best option is the 192-bit curve since about 12 % of energy can be saved.

Additionally to the ECDH, the introduction of digital signature brings more security in terms of access control. For this case, the NFC-I verifies the digital signature of the smart phone and if the signature is not valid, the NFC-I does not allow the communication to the target device. However, the introduction of the digital signature produces a higher

energy consumption of about 20 %.

Finally to give a raw idea of how much energy a ECDH needs in comparison with background services, one could say that about 18 % more energy has to be invested. As background services can be considered the backlight of the display, WIFI, operation system processes, the transfer power of NFC etc.

Chapter 7

Future Work

This master thesis can be improved by adding additional security in different components of the system:

- *Smart phone:* The smart phone is not really a secure device, like in every operating system there is the possibility to install sky software to read out secure data. The use of the cryptography on the secure element should solve this problem. A problem to consider, is that, the communication from the smart phone to the secure element has to be protected again.
- *Target device:* The system bus (AMBA) is protected by cryptography, the problem is that the used key is hard coded in both components. It would be make more sense to think about a way to change regularly the key in both components. Possible proposals:
  - One side is the master and generates a key with a PRNG and sends it encrypted with the old key to the slave. In the next data transmission both sides use the new distributed key.
  - Both sides has a look up table with keys which are used one after the other.
  - Both sides has integrated the same algorithm to generate the same key.
- *Smart phone and NFC-I:* Introduce digital signature also from the NFC-I back to the smart phone to avoid man-in-the-middle attacks.

Another possible improvement would be to make the security system usable for a wider range of use cases. In this case the smart phone can be pre-loaded with different target device configurations and depending on which communication occurs, the correct configuration is loaded automatically as well as the defined security level etc. In this case, it would be possible to use the smart phone for all possible interactions like, for example, to open the house's door. This use case would need a higher security level than controlling the sun-bland.

Appendix A

Bouncy castle supported curves for ECDSA

A.1 Fp

X9.62	
Curves	Size (in bits)
prime192v1	192
prime192v2	192
prime192v3	192
prime239v1	239
prime239v2	239
prime239v3	239
prime256v1	256
SEC	
Curves	Size (in bits)
secp224r1	224
secp256r1	256
secp384r1	384
secp521r1	521
NIST (aliases for SEC curves)	
Curves	Size (in bits)
P-224	224
P-256	256
P-384	384
P-521	521

Table A.1: Supported curves over Fp [cas].

A.2 F2m

X9.62	
Curves	Size (in bits)
c2pnb163v1	163
c2pnb163v2	163
c2pnb163v3	163
c2pnb176w1	176
c2tnb191v2	191
c2tnb191v1	191
c2tnb191v3	191
c2pnb208w1	208
c2tnb239v1	239
c2tnb239v2	239
c2tnb239v3	239
c2pnb272w1	272
c2pnb304w1	304
c2tnb359v1	359
c2pnb368w1	368
c2tnb431r1	431
SEC	
Curves	Size (in bits)
sect163r2	163
sect233r1	233
sect283r1	283
sect409r1	409
sect571r1	571
NIST (aliases for SEC curves)	
Curves	Size (in bits)
B-163	163
B-233	233
B-283	283
B-409	409
B-571	571

Table A.2: Supported curves over F2m [cas].

Appendix B

Parameter of the used elliptic curves

B.1 secp160r1 [Res00]

p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 7FFFFFFF

a = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 7FFFFFFC
b = 1C97BEFC 54BD7A8B 65ACF89F 81D4D4AD C565FA45

G = 4A96B568 8EF57328 46646989 68C38BB9 13CBFC82
G_x = 4A96B568 8EF57328 46646989 68C38BB9 13$CBFC$82
G_y = 23A62855 3168947D 59DCC912 04235137 7AC5FB32

n = 00000000 00000000 0001$F4C$8 F927AED3 CA752257
h = 01

B.2 secp192r1 [Res00]

p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF

a = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFC
b = 64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1

G = 188DA80E B03090F6 7CBF20EB 43A18800 F4FF0AFD 82FF1012
G_x = 188DA80E B03090F6 7CBF20EB 43A18800 F4FF0AFD 82FF1012
G_y = 07192B95 FFC8DA78 631011ED 6B24CDD5 73F977A1 1E794811

n = FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831
h = 01

B.3 secp224r1 [Res00]

p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 00000000 00000001

a = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFE
b = B4050A85 0C04B3AB F5413256 5044B0B7 D7BFD8BA 270B3943 2355FFB4

G = B70E0CBD 6BB4BF7F 321390B9 4A03C1D3 56C21122 343280D6 115C1D21
G_x = B70E0CBD 6BB4BF7F 321390B9 4A03C1D3 56C21122 343280D6 115C1D21
G_y = BD376388 B5F723FB 4C22DFE6 CD4375A0 5A074764 44D58199 85007E34

n = FFFFFFFF FFFFFFFF FFFFFFFF FFFF16A2 E0B8F03E 13DD2945 5C5C2A3D
h = 01

Appendix C

FPGA utilization summary

C.1 Spantan 3 (3s1500fg346-4) FPGA utilization summary

Component	Total available amount	Used amount	[%]
Slices	13313	10256	77
Slices Flip Flop	26624	6570	24
4 input LUTs	26624	19546	73
Bounded IOs	333	185	55
Block RAM	32	20	62
Global Clocks	8	7	87
Digital clock manager	4	4	100

Table C.1: FPGA utilization summary.

Appendix D

Acronyms

AES Advance Encryption Standard

ANSI American National Standards Institute

APDU Application Protocol Data Unit

API Application Programming Interface

CA Certificate Authority

CPU Central processing unit

DES Data Encryption Standard

DSA Digital Signature Algorithm

ECC Elliptical Curve Cryptography

ECDH Elliptic curve Diffie–Hellman

ECDSA Elliptic curve digital signature algorithm

FIFO First In First Out

FPGA Field-programmable gate array

GPS Global Positioning System

ID Identification Data

LUT Look-up table

MD4 Message-Digest Algorithm 4

MD5 Message-Digest Algorithm 5

NFC Near Field Communication

NFC-I Near Field Communication-Interface

PDA Personal Digital Assistant

PIN Personal identification number

PRNG Pseudo Random Number Generator

RF Radio Frequency

RFID Radio-frequency identification

RIPEMD RACE Integrity Primitives Evaluation Message Digest

SD Secure Digital

SHA-1 Secure Hash Algorithm - 1

SMS Short Message Service

SoC System on Chip

TPM Trusted Platform Module

WBAN Wireless Body Area Network

Bibliography

[ADF07] M. Aigner, S. Dominikus, and M. Feldhofer. A system of secure virtual coupons using NFC technology. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, pages 362–366. IEEE, 2007.

[AJMV01] Paul C. van Oorschot Alfred J. Menezes and Scott A. Vanstone. *Handbook of Applied Cryptography*. 2001.

[BGS+10] Christian Bachmann, Andreas Genser, Christian Steger, Reinhold Weiss, and Josef Haid. Automated Power Characterization for Run-Time Power Emulation of SoC Designs. In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pages 587–594. IEEE, 2010.

[BKK+09] J.W. Bos, M.E. Kaihara, T. Kleinjung, A.K. Lenstra, and P.L. Montgomery. On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography. Technical report, Cryptology ePrint Archive, Report 2009/389, 2009.

[Bou] The Legion of the Bouncy Castle. http://www.bouncycastle.org/.

[BP09] Nathan Beckmann and Miodrag Potkonjak. Hardware-based public-key cryptography with public physically unclonable functions. In Stefan Katzenbeisser and Ahmad-Reza Sadeghi, editors, *Information Hiding*, volume 5806 of *Lecture Notes in Computer Science*, pages 206–220. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-04431-1_15.

[BSS99] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.

[cas] Bouncy castle. Supported Curves (ECDSA and ECGOST). http://www.bouncycastle.org/wiki/display/JA1/Home.

[DA07] S. Dominikus and M. Aigner. mCoupons: An Application for Near Field Communication (NFC). In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 2, pages 421–428. IEEE, 2007.

[DBP96] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A strengthened version of RIPEMD. In Dieter Gollmann, editor, *Fast Software*

Encryption, volume 1039 of *Lecture Notes in Computer Science*, pages 71-82. Springer Berlin / Heidelberg, 1996. 10.1007/3540608656_44.

[DK07] H. Delfs and H. Knebl. *Introduction to Cryptography: Principles and Applications*. Information Security and Cryptography. Springer, 2007.

[DR02] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[Ell09] The Case for Elliptic Curve Cryptography, 2009. http://www.nsa.gov/business/programs/elliptic_curve.shtml.

[GAI] AEROFLEX GAISLER. Leon 3. http://www.gaisler.com/.

[HB06] E. Haselsteiner and K. Breitfuß. Security in near field communication (NFC). In *Workshop on RFID Security RFIDSec*, volume 6, 2006.

[hit] http://www.hitachi.com/rd/yrl/people/pki/index04.html.

[HLSS06] O. Henniger, K. Lafou, D. Scheuermann, and B. Struif. Verifying X. 509 Certificates on Smart Cards. In *International Conference on Computer and Information Science and Engineering (CISE)*, 2006.

[HMV03] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[HT10] M. Hutter and R. Toegl. A Trusted Platform Module for Near Field Communication. In *Systems and Networks Communications (ICSNC), 2010 Fifth International Conference on*, pages 136–141. Ieee, 2010.

[HT11] M. Hutter and R. Tögl. Touch'n Trust: An NFC-Enabled Trusted Platform Module. *International Journal On Advances in Security*, 4(1 and 2):131–141, 2011.

[ILR] H. IVEY-LAW and R. ROLLAND. Finding Cryptographically Strong Elliptic Curves: A Technical Report.

[Inf11] Infineon. Infineon Chip Card and Security ICs Portfolio, November, 2011.

[key] Elliptic Curves in Public Key Cryptography: The Diffie Hellman Key Exchange Protocol and its relationship to the Elliptic Curve Discrete Logarithm Problem. http://ocw.mit.edu/courses/mathematics/18-704-seminar-in-algebra-and-number-theory-rational-points-on-elliptic-curves-fall-2004/projects/haraksingh.pdf.

[Kil09] M. Kilas. Digital Signatures on NFC Tags. *Unpublished master's thesis, KTH Royal Institute of Technology, Sweden*, 2009.

[Kob87] N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[KV01] Henry Kuo and Ingrid Verbauwhede. Architectural Optimization for a 1.82Gbits/sec VLSI Implementation of the AES Rijndael Algorithm. In *Implementation of the AES Rijndael Algorithm, CHES 2001, LNCS 2162*, pages 51–64. Springer, 2001.

[Mil86] Victor Miller. Use of elliptic curves in cryptography. In Hugh Williams, editor, *Advances in Cryptology - CRYPTO 85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin / Heidelberg, 1986.

[MLKS08] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger. NFC Devices: Security and Privacy. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 642–647. Ieee, 2008.

[MM] Manuel Trebo Fioriello Manuel Menghin, Norbert Druml. PtNFCGate - A power-aware and trusted Near Field Communication Gateway to embedded systems.

[Ope01] OpenSSL Project, 2001. http://www.openssl.org.

[oST] Information Technology Laboratory National Institute of Standards and Technology. Secure Hash Standard (SHS). FIPS PUB 180-4.

[oST07] National Institute of Standards and Technology. Recommendation for Key Management - Part 1: General, March, 2007.

[PMW10] M. Potkonjak, S. Meguerdichian, and J.L. Wong. Trusted sensors and remote sensing. In *Sensors, 2010 IEEE*, pages 1104 –1107, nov. 2010.

[PRRJ06] N.R. Potlapally, S. Ravi, A. Raghunathan, and N.K. Jha. A Study of the Energy Consumption Characteristics of Cryptographic Algorithms and Security Protocols. *Mobile Computing, IEEE Transactions on*, 5(2):128 – 143, feb. 2006.

[Rej] Basagic Rejhan. NFC Communication between Android based Smart Phones and NFC Interface equipped Devices. Master's thesis.

[Res00] Certicom Research. SEC 2: Recommended Elliptic Curve Domain Parameters, 2000. version 1.0.

[res11] IMS research. NFC, Secure Payments and Digital Security - A Market Summary, April 2011.

[Riv92a] R. Rivest. The MD4 Message-Digest Algorithm, RFC 1320, 1992.

[Riv92b] R. Rivest. TThe MD5 Message-Digest Algorithm. 1992.

[RLS12] M. Roland, J. Langer, and J. Scharinger. Practical Attack Scenarios on Secure Element-Enabled Mobile Devices. In *Near Field Communication (NFC), 2012 4th International Workshop on*, pages 19 –24, march 2012.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[Sch85] R. Schoof. Elliptic Curves Over Finite Fields and the Computation of Square Roots mod p. *Math. Comp*, 44(170):483–494, 1985.

[SSPK12] Jacob M. Sorber, Minho Shin, Ron Peterson, and David Kotz. Plug-n-Trust: Practical Trusted Sensing for mHealth. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, pages 309–322, New York, NY, USA, 2012. ACM.

[Sta95] Federal Information Processing Standard. Secure hash standard, April 1995. FIPS-180-1.

[tOSITD00] Information technology Open Systems Interconnection The Directory. Public-key and Attribute Certificate Frameworks, 2000.

[VDWKP09] Gauthier Van Damme, Karel M. Wouters, Hakan Karahan, and Bart Preneel. Offline NFC Payments with Electronic Vouchers. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, MobiHeld '09, pages 25–30, New York, NY, USA, 2009. ACM.

[VDWP09] G. Van Damme, K. Wouters, and B. Preneel. Practical Experiences with NFC Security on mobile Phones. In *Workshop on RFID Security–RFIDSec'09*, 2009.

[WGE+05] A.S. Wander, N. Gura, H. Eberle, V. Gupta, and S.C. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 324 – 328, march 2005.

[Wol] Ruben Wolf. Verifikation digitaler Signaturen. http://www.informatik.tu-darmstadt.de/BS/Lehre/Sem98_99/T11/index.html.

[WYY05] X. Wang, Y. Yin, and H. Yu. Finding Collisions in the Full SHA-1. In *Advances in Cryptology–CRYPTO 2005*, pages 17–36. Springer, 2005.