Master's Thesis

Design and Implementation of a Controller-based Brand Protection based on Elliptic Curves

Adnan KULETA, BSc

Institute for Technical Informatics
Graz University of Technology
Austria

Assessor: Ass.Prof. Dipl.-Ing. Dr. techn. Christian Steger
Advisor: Ass.Prof. Dipl.-Ing. Dr. techn. Christian Steger
Dipl.-Ing. Nobert Druml, BSc

Graz, May 2014

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

In the name of God the most Beneficent and the most Merciful, the one who rules all over the earth, heavens and everything that is present in between. All praise is to God for all His power, His blessings, His majesty, His sovereignty and every thing related to Him, whether we know it or not. Peace and prayer be upon our beloved Prophet Muhammad (S.A.W), his family and all of his companions.

This master thesis has been performed in 2014 at the Institute for Technical Informatics, Technical University in Graz in cooperation with Enso Detego GmbH, specialized on RFID software solution.

I would like to sincerely and greatfully thank Dipl.-Ing. Norbert Druml, Dipl.-Ing. Manuel Menghin and Ass.Prof. Dipl.-Ing. Dr. techn. Christian Steger at Institute for Technical Informatics for their guidance, understanding and professional advices.

I would like also to thank Dipl.-Ing. Matthias Weitlaner, Dipl.-Ing. (FH) Alexander Krenn, and the whole team at Enso Detego GmbH for their great and friendly support. I would like to thank also the Institute for Applied Information Processing and Communications (IAIK) at the Technical University in Graz, for providing me some test data. Also, I like to thank Mrs. Andreas Höller for using part of her master's thesis implementation as a reference in this project.

Finally, I would like to thank my sweet family for their aid during my studying and all my friends for a great cooperation.

Graz, May 2014 Adnan A. KULETA

Contents

List of Figures

List of Abbreviations

AES Advanced Encryption Standard
APDU Application Protocol Data Unit
API Application Programming Interface
ANSI American National Standards Institute
C-APDU Command Application Protocol Data Unit
CPU Central Processing Unit
CRT Certificate Revocation List
DH Diffie-Hellman
DES Data Encryption Standard
DLP Discrete Logarithm Problem
DSA Digital Signature Algorithm
DVM Dalvik Virtual Machine
ECC Elliptic Curve Cryptography
ECDH Elliptic Curve Diffie-Hellman
ECDSA Elliptic Curve Digital Signature Algorithm
ECDHP Elliptic Curve Diffie-Hellman Problem
GC Garbage Collector
GDLP Generalized Discrete Logarithm Problem
HAL Hardware Abstraction Layer
HF High Frequency
Hz Hertz
ID Identification
IEC International Electrotechnical Commission
IEEE Institute of Electrical and Electronics Engineers
IFF Identification Fiend or Foe
ISO International Organization for Standardization
IPSec Internet Protocol Security
JDK Java Development Kit
kHz Kilohertz
LF Low Frequency
MIME Multiple Internet Mail Extensions
MITM Man-in-the-Middle
MHz Megahertz
NDEF NFC Data Exchange Format
NFC Near Field Communication
NIST National Institute of Standards and Technology

OHA Open Handset Alliance
P2P Peer-to-Peer
PCD Proximity Coupling Devices
PICC Proximity Cards or Objects
PTF Power Transfer Function
R-APDU Response Application Protocol Data Unit
RFID Radio Frequency Identification
RF Radio Frequency
RSA Rivest-Schamir-Adlemant
RTD Record Type Definition
SLE System of Linear Equations
SSH Secure Shell
TLS Transport Layer Security
TOE Target of Evaluation
UID Unique Identification
UHF Ultra High Frequency
NAF Non-Adjacent Form
wNAF window Non-Adjacent Form

Abstract

As postulated by Gordon Moore, the complexity and performance of integrated circuits and embedded devices is still increasing exponentially. Integration of wireless technology in embedded devices such as mobile devices affected customer's life in a positive way. Customers do not use mobile devices only for telephony but also for payment, access, and administration of non-wireless equipment. Those and many other functions are provided by the integration of Near Field Communication (NFC) in embedded devices. In the usual applications, the detection range of NFC is limited to only a few centimeters. However attacks on NFC are always possible. Adversaries may try to interrupt the communication or even steal or modify the properties of the RFID tag.

If the tag is used as a token against counterfeiting, cloning the tag may also be in the adversary's interest. Integrating security features like Elliptic Curve Cryptography (ECC) can successfully mitigate such attacks. The design and implementation of ECC in a RFID tag is a challenging task, due to chip constraints.

This master's thesis presents a very efficient way to avoid the cloning of an RFID tag by the integration of an optimized ECC-based Diffie-Hellman key exchange, known as one way authentication protocol. From a complexity-theoretical point of view, the solution of a System of Linear Equations (SLE) is efficiently computable. As the hardware implementation which allows us to perform ECC calculations over prime fields more efficiently than over binary fields. However, for a good performance only a single multiplication is applied on the Infineon security controller which resembles the RFID tag.

The first task for this protocol is a software simulation of an optimized ECC-based authentication protocol for security controller-based applications. After that, a communication between an NFC-enabled smart phone and the security controller is realized. Here the reference simulation of multiplication is applied on the security controller by using Application Protocol Data Unit (APDU). An evaluation of Android-based crypto libraries and security controller's crypto library is applied regarding applicability for the authentication protocol. The next task is the implementation of a standard but unoptimized ECC-based authentication protocol on NFC-enabled smart phone and security controller. The result, responded from the security controller, is compared with simulations results, and in case of a correctness a valid authentication is achieved. In this way, the authenticated tags and readers will successfully communicate to each other. A comparison between an optimized and unoptimized ECC authentication protocol is performed with the help of Android libraries.

Finally, the and performance behavior of the reader and the tag with this authentication protocol is evaluated which shows faster results comparing to approaches in related work.

Kurzfassung

Wie von Gordon Moore postuliert, erhöht sich die Komplexität und Leistung integrierter Schaltungen und eingebettetes System immer noch exponentiell. Die Integration der 'Wireless'-Technologie in eingebettetes System, wie mobilen Geräten beeinflussen das Leben des Kunden in positiver Art und Weise. Kunden verwenden mobile Geräte nicht nur zum Telefonieren, sondern auch für Zahlung, sowie Zugriff und Verwaltung von Nicht-Wireless-Geräten.
Diese und viele andere Funktionen werden durch die Integration der Near-Field-Kommunikation (NFC) in eingebettetes System bereitgestellt. In normalen Anwendungen ist die Erkennungsreichweite von NFC auf wenige Zentimeter beschränkt. Trotzdem sind Angriffe auf NFC immer möglich. Gegner könnten versuchen die Kommunikation zu unterbrechen, bzw. sogar die auf dem RFID Tag gespeicherte Information zu stehlen oder zu verändern. Wenn der Tag als Fälschungsschutz verwendet wird, kann das Klonen des Tags ebenfalls im Interesse des Gegners sein. Die Integration von Sicherheitsalgorithmen wie Elliptic Curve Cryptography (ECC), können solche Angriffe erfolgreich abwehren. Entwurf und Implementierung von ECC auf einem RFID Tag sind wegen Chip-spezifischen Einschränkungen eine herausfordernde Aufgabe.
In dieser Masterarbeit wird ein sehr effektiver Weg beschrieben, um das Klonen von RFID Tags durch die Integration eines optimierten ECC-basierten Diffie-Hellman Schlüsselaustausches zu verhindern. Von einem komplexen theoretischen Gesichtspunkt aus betrachtet, sind lineare Gleichungssysteme (SLE) effizient berechenbar, während die Hardware-Implementierung es uns erlaubt ECC. Berechnungen über prim Felder effizienter durchzuführen als über binäre Felder. Um jedoch eine gute Leistung zu erzielen, wird nur eine einzige Multiplikation auf dem Security-Controller, bekannt als RFID Tag, ausgeführt.
Der erste Teil dieser Arbeit befasst sich mit der Software Simulation der Montgomery Multiplikation. Danach wird die Kommunikation zwischen einem NFC-fähigen Smartphone und einem Security-Controller realisiert, bei dem die Referenz-Simulierung der Multiplikation auf dem Security-Controller unter Verwendung der Application Protocol Data Unit (APDU) durchgeführt wird. Eine Evaluierung von Android-basierten und Security-Controller Krypto-Bibliotheken wird durchgeführt bzw. für das Authentisierungs-Protokoll angewandt. Entwurf und Implementierung eines Standard- aber nicht optimierten ECC-basierten Authentisierungs-Protokolls auf NFC-fähigen Smartphones und Security-Controller/Tag sind erfolgt. Das Ergebnis, resultierend aus dem Security Controller wird mit Simulationsergebnissen verglichen und im Falle der Übereinstimmung eine gültige Authentisierung erreicht. Deshalb können authentisierte Tags und Lesegeräte erfolgreich miteinander kommunizieren. Ein Vergleich zwischen einem optimierten und nicht optimierten ECC Authentisierungs-Protokoll wird unter Verwendung von Android-Bibliotheken,

eigener Implementierung, usw. durchgeführt.
Abschliessend werden Zeitanalysen der Lesegeräte und Tags für dieses Authentisierungs-Protokoll durchgeführt, welche sehr gute Ergebnisse verglichen mit ähnlichen Arbeiten liefern.

Chapter 1

Introduction

The tremendously rapid growth of the technology impacted incredibly the life of the people, starting from automotive industry, televisions towards to embedded devices such as mobile phones. There is a less percentage usage of mobile phones only for calling. The majority of the people use their mobile phones also for different purposes for example reading emails, organizing their tasks, finding an address etc. These are the lowly functionalities that people at least can do with their own mobile phones nowadays. Everyone wants to have as less things as possible with themselves such as documents, cards etc. Near Field Communication (NFC) the new technology integrated into embedded devices especially smart phones, replaces not even papers but also a wallet with various cards e.g. credit cards, access cards etc. Due to this positive and helpful interface provided by smart phones, also the usage of smart phones by the customers increased too.

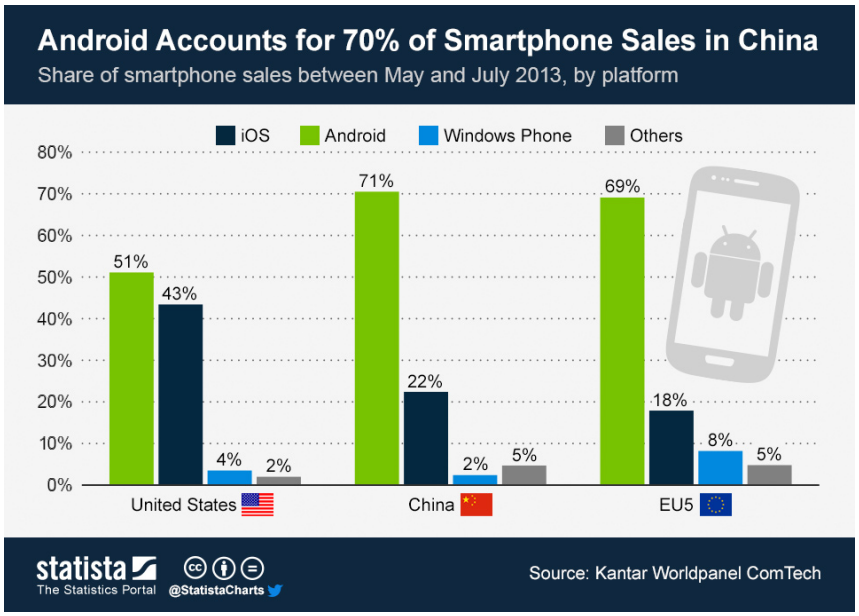


Figure 1.1: Android OS Smart Phone Sale in China [RS]

Only in the third quarter of 2013 there were sold 455.6 million smart phones around the world, which is an increase of 5.7 percent from the same period 2012 [MR]. According to Gartner, Inc. [MR] only in the third quarter of the year 2013 accounted for 55 percent of overall sales of smart phones. From the point of view as the most successful NFC-enabled smart phone manufacturer in the year 2013 are Android based OS smart phones. Figure 1.1 shows that 70 % of the sold smart phones in China between May and July were Android based smart phones. According to Juniper Research [Mur] report analysed in a year of 2011, that 1 in 5 smart phones will be NFC-enabled by the year 2014. This new technology (NFC) is offering a tremendous efforts. But the main question from the users of NFC-enabled smart phones still remains: **IS THIS A SECURY TECHNOLOGY THAT CONVINCES US TO STORE EVERYTHING ON THE SMART PHONE?** This is a very interesting question but difficult to answer. I would say that "*there is nothing secure nowadays in the world*". Attackers try always to find a lack, and after that do an eavesdropping, modifying data etc. NFC-enabled smart phones have a distance of a detection of less than 10 cm, so how is it possible to attack here?

1.1 Motivation and Goal

The main problem that we are facing nowadays with the NFC-enabled smart phones is not only attacking on the smart phones, but also cloning of a Radio Frequency Identification (RFID) tag as shown in the Figure 1.2. This means that the same RFID tag is cloned

Figure 1.2: RFID Tag Cloning (adapted from [BBD+])

multiple times, and only one of them is the original RFID tag, the others are counterfeited without the knowledge of an authorized owner of the original RFID tag. In the case of a protection, RFID signals must be encrypted, which is quite expensive, round $5 costs a secured tag, and due to this prize, most of the commercial RFID tags do not include security [Mol]. For instance, the chips that are integrated into US passports are very good secured, which makes them very difficult for unauthorized readers to retrieve information stored on it (person's name, age, nationality etc.) [Mol].

The aim of this master thesis is to prevent the cloning of an RFID tag, through the implementation of an authentication protocol. Authentication is one out of many methods to protect against counterfeiting of an entity. An optimized ECC-based authentication protocol is applied between an Android NFC-enabled smart phone and an RFID tag (which uses a security controller) as shown in the Figure 1.3. An algorithm applied within an optimized ECC-based authentication protocol is the Montgomery multiplication algorithm defined over prime field. This is an efficient algorithm which computes modular multiplications without using modular expensive divisions. Therefore, it is widely used for

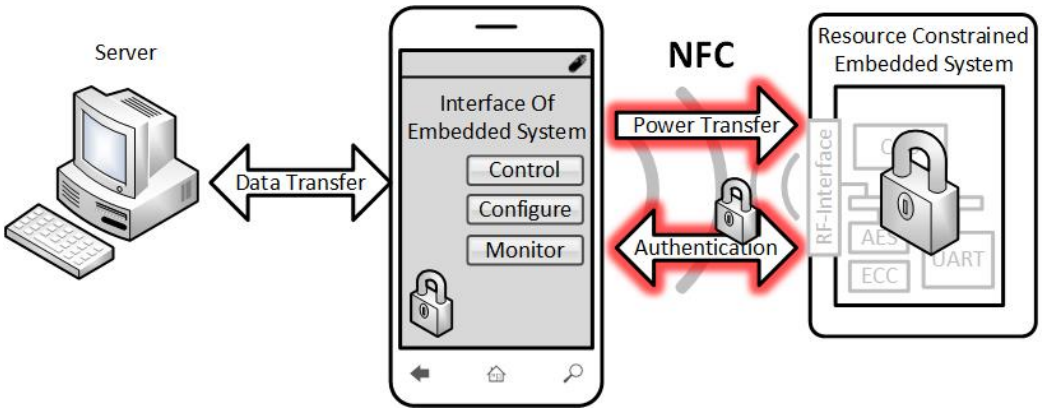


Figure 1.3: Authentication Protocol system overview

public-key cryptosystems, and is suitable for constrained embedded devices memory e.g. smart cards. Furthermore, the security controller used in this master thesis also supports System of Linear Equations (SLE) multiplier and is optimized for elliptic curves defined over prime fields. A software timing analysis of the Montgomery multiplication algorithm is evaluated as a part of the authentication protocol. Furthermore, a timing analysis of the interaction of the NFC-enabled smart phone and the security controller is carried out.

This master's thesis is a part of the META[:SEC:] (**M**obile **E**nergy-efficient **T**rustworthy **A**uthentication **S**ystems with **Elliptic Curves** based **SEC**urity) project as shown in the Figure 1.4. META[:SEC:] is realized in cooperation with Institute for Technical Informatics[1], Infineon Technologies Austria AG[2] and Enso Detego GmbH[3] which supported financially my master's thesis. The main focus of the META[:SEC:] project is the evaluation of the power and security on the system level within a trusted system development, an optimized ECC-based authentication protocol takes place at the **Security and Fault Concepts**. This project investigates, optimizes and implements an ECC-based architecture for the mobile-security applications. Further information about META[:SEC] project is available online at [SMD]

1.2 Outline

This master's thesis is organized as follows. Chapter 2 outlines the related work, where Sections 2.1 and 2.2 describe the basics about RFID and NFC, whereas for more information, some books and scientific papers are listed within these sections. Section 2.3 provides the system architecture of Android OS. Section 2.4 gives a general overview about the types of cryptography: symmetric 2.4.1 and asymmetric 2.4.2. The main reason of using ECC in this master's thesis, and its mathematical equations for elliptic curve defined over prime field is described in the Section 2.4.3. Section 2.5 gives a brief overview about various projects related to our work and they can be used further to compare with our obtained

[1]http://www.iti.tugraz.at/
[2]http://www.infineon.com/
[3]http://www.enso-detego.com/

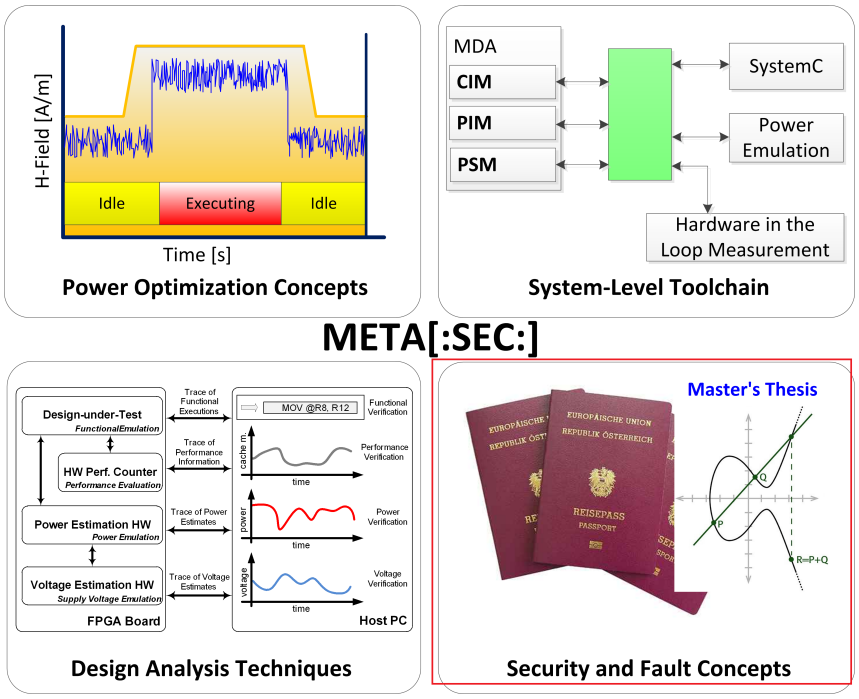


Figure 1.4: META[:SEC:] project overview

results. Chapter 3 describes in details the design specifications of this master's thesis. The design of a software simulation for an authentication protocol is provided in the Section 3.4.1. The connection establishment between an Android NFC-enabled smart phone and the security controller is described in Section 3.1. The field arithmetic operations defined over the prime field are elaborated in the Section 3.2 and the Montgomery Multiplication Algorithm defined over prime field is described in the Section 3.3. Section 3.4 gives an overview about an optimized one-way authentication protocol. Chapter 4 shows the implementation platform of this authentication protocol on an Android NFC-enabled smart phone. Results obtained for the optimized ECC-based authentication protocol are described and compared in Chapter 5. A conclusion about this master's thesis is described in the Chapter 6, and finally future work is shown in the Chapter 7.

Chapter 2

Related Work

This chapter covers the basic components of Radio Frequency Identification (RFID), Near Field Communication (NFC), Android OS and Elliptic Curve Cryptography (ECC). Furthermore, similar projects are described and used as an idea for the design and implementation of this Thesis.

2.1 Radio Frequency Identification

Radio Frequency Identification (RFID) is a system which is used to wirelessly describe the identity of an object or person, through the use of radio waves [Vio]. It was used for the first time by the United Kingdom during the World War II in case of identifying an aircraft as friend or foe [Rob]. It scans a tag or label which is attached to any object as an identifier and reads the data stored on it. RFID is used to identify people, goods, animals etc. Further information on RFID is available in the book *RFID Handbook, Fundamentals and Application in Contacless Smart Cards, Radio Frequency Identification and Near-Field Communication* [Fin10].

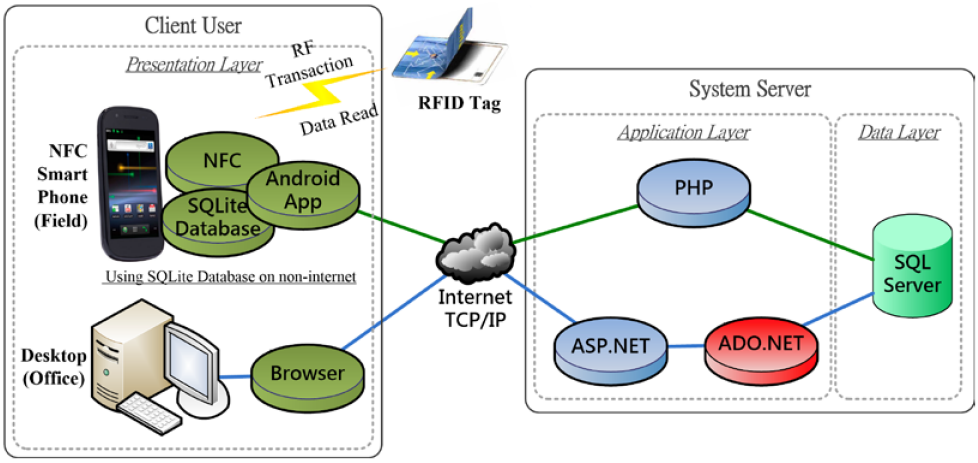


Figure 2.1: HF-Band RFID-System [LSL+]

RFID operates at different frequency bands [Fin10]:

- **Low Frequency (LF):** operates in 125-134 kHz frequency band
- **High Frequency (HF):** operates at 13.56 MHz and is the frequency band that used in NFC (see Section 2.2)
- **Ultra High Frequency (UHF):** operates in 865-868 MHz frequency band

As shown in Figure 2.1 HF-Band RFID system consists of some of the following components [Fin10]:

- Tags (see Section 2.1.2)
- Reader (see Section 2.1.1)

2.1.1 RFID Readers

RFID readers emit readable messages stored in tags over an air interface. An RFID reader sends energy to the tag and tag responds back depending on the request sent from the reader. Communication between reader and tag may be a simple ping or more complex multi-round protocol, it is important to secure the communication by performing *anti-collision* protocol [Fin10]. Readers operate at different frequencies and store data locally. An RFID reader is connected to a back-end database server, where the information read from a tag is stored. RFID readers are also nowadays integrated into mobile devices (see Sections 2.2 2.3).

2.1.2 RFID Tags

Tags or labels are attached to an object to be identified and contain information about this object. RFID tags are divided into two groups: passive and active or battery-assisted. The main difference between the two groups of tags is that a passive tag is cheaper and an operation can be started if a reader sends energy to a passive tag. If a tag is within the range of reader, it is powered.

Tags may be read-only which have a serial key and read/write which can be written by the system user.
The description of the other components of HF-Band RFID system is available at [LSL+]

2.1.3 Power awareness and security aspects in HF-Band RFID Systems

A HF-Band RFID system is mainly used in embedded devices where its functionality is increasing e.g. payment, identification, etc. Even though high energy consumption is necessary for this kind of functionality, the battery life of embedded devices still remains as a critical consideration. Therefore different techniques for power management are implemented although there is still a certain amount of energy wastage whilst communication between HF-Band RFID reader and transponder takes place. Menghin et al. [MDS+13] elaborated magnetic field strength scaling as one of the best techniques of reducing the amount of wasted energy. Menghin et al. elaborated three different investigations into the

usage of field strength scaling to prevent the waste of energy. One of these investigations is Power Transfer Function (PTF)-Determinator method applied at card detection phase. PTF-method was integrated in an application to read digital business cards during runtime. The simulation results of their work show if field strength is applied then up to 26 % less transmission energy (energy drain of NFC) is needed and the smart phone's battery drain can be decreased up to 13 %.

Another very important aspect of HF-Band RFID system is the security of this technology.
Knospe et al. [KP04] described the security properties of RFID system as follows:

- **Confidentiality**: In most cases communication between reader and tag is unprotected (exception of some high-end ISO 14443 systems).
- **Integrity**: Checksums (CRC) are often applied within the communication for the purpose of integrity assurance.
- **Availability**: Communication interruption between an RFID reader and tag can be exploited by an RFID Blocker.
- **Authenticity**: The cloning of a tag is a risk nowadays and therefore the main point of this master's thesis is the design and implementation of an optimized authentication protocol.
- **Anonymity**: A person or an object carring a tag can be traced without being noticed. This occurs most in supply-chain applications where tags are read automatically and permits the counting of undesired objects.

2.2 Near Field Communication

Near Field Communication (NFC) is based on RFID technology and operates at short-range up to a distance of 10 cm of wireless connectivity. NFC operates in the globally available unlicensed Radio Frequency (RF) band of 13.56 MHz and data rate transfer varies from 106 to 424 kbit/s. Furthermore, NFC communication is achieved by triggering two NFC-enabled devices that are close to each other. NFC standards are based on existing Radio Frequency Identification (RFID) (see Section 2.1) for **ISO/IES 14443** [PBJ06] Type A and Type B [Cor05] and **FeliCa**[1]. The NFC Forum was founded in 2004 by Nokia, Philips Semiconductors (which in 2006 became NXP Semiconductors[2]) and Sony and has more than 170 members [For13a].

The integration of NFC into embedded devices provides not only one-way communication but also two-way communication between endpoints.

2.2.1 NFC Standard Specifications and Communication Mode

As mentioned at the beginning, NFC is a short-range wireless communication. Table 2.1 shows the comparison between NFC and some other wireless technologies. As listed, NFC

[1] http://www.sony.net/Products/felica/about/
[2] www.nxp.com/

has some similar characteristics to other wireless technologies for example RFID but completely different characteristics to Bluetooth and Infrared.

Table 2.1: Comparison of NFC to other wireless technologies (adapted from [Asc13])

	NFC	RFID	IrDa	Bluetooth
Set - up time	<0.1 ms	<0.1 ms	$\sim 0.5\,s$	$\sim 6\,sec$
Range	Up tp 10 cm	Up to 3 m	Up to 5 m	Up to 30 m
Usability	Human centric Easy, intuitive fast	Item centric Easy	Data centric Easy	Data centric Medium
Selectivity	High, given, security	Partly given	Line of sight	Who are you?
Use cases	Pay, get access, share, initiate service, easy setup	Item tracking	Control and exchange data	Network for data exchange, headset
Consumer experience	Touch wave, simply connect	Get information	Easy	Configuration needed

NFC Terminology

NFC Forum devices support the following ISO/IEC 14443 Parts:

- **Physical Characteristics** [14408a] defines the size and physical characteristics of the card known as proximity cards (PICCs).
- **Radio frequency and signal interface** [14410] specifies the characteristics of the field which must be provided for power of communication between proximity coupling devices (PCDs) and proximity cards.
- **Initialization and anticollision** [14401] describes
  1. polling of PICCs entering the field of PCD
  2. byte format, timing and frames used during communication between PICCs and PCD
  3. request and answer to request
- **Transmission protocol** [14408b], specifies a half-duplex transmission protocol which is also further used with other parts of ISO/IEC 14443.

NFC supports four types of tags which are operable with NFC devices [For13b].

NFC also supports MIFARE contactless smartcards [MIF13]. MIFARE is the NXP Semiconductors-owned trademark and based on ISO/IEC 14443 Type A contactless smart cards. Table 2.2 shows the support of MIFARE contactless cards in standard ISO/IEC 14443. MIFARE covers different kinds of contactless cards [MIF].

Table 2.2: MIFARE and ISO 14443 (adapted from [PBJ06])

	MIFARE® Ultralight	MIFARE® Standard	MIFARE® Desfire
ISO/IEC 14443 part 1	yes	yes	Yes
ISO/IEC 14443-A part 2	yes	yes	Yes
ISO/IEC 14443-A part 3	yes	yes	Yes
ISO/IEC 14443 part 4	No, fixed command set	No, fixed command set	Yes

Data exchange between two NFC Forum devices or NFC Forum device and a passive NFC Forum tag is called NFC Data Exchange Format (NDEF) message [Inc06]. NDEF message can be composed from one or more NDEF records. NDEF message contains an array of NDEF records. An NDEF record holds a header and a payload. It is important to mention that the number of NDEF records depends on the type of application and NFC Forum tag. NFC Forum defines these types of information as Record Type Definition (RTD) and provides different variatons of them [For13b]:

- **TextRecord RTD:** Stores a text strings in multiple languages.
- **URI RTD:** It stores a Uniform Resources Identifiers (URI).
- **Smart Poster RTD:** It is used to put URLs, SMSs on an NFC Forum tag or transport them between NFC-enabled devices.
- **Generic Control RTD:** It provides a specific request such as starting of application.
- **Signature RTD:** It provides a list of signature algorithms that can be used to create a signature, a format used when signing single or multiple NDEF records.

Figure 2.2: NFC Communication: Reader/Writer mode [LTD]

NFC Forum defines three different communication modes:

Read/Write mode allows applications for the transmission of NFC Forum-defined messages. This form of communication allows to write to NFC or many contactless cards. For example, if an NFC tag is attached to a poster, the information stored on its tag can only be read by an NFC-enabled device by tapping on the chip as shown on Figure 2.2.

Figure 2.3: NFC Communication: Peer to Peer mode [LTD]

Peer-to-Peer mode is defined as device to device link-level communication and data exchange between them as shown on Figure 2.3. In the case of a large amount of data needing to be exchanged, peer to peer mode can also be used for a high data speed rate through connection establishment over Bluetooth or WiFi.

Figure 2.4: NFC Communication: Card Emulation Mode [LTD]

Card Emulation mode allows the NFC-handset to behave as a standard smart card or a transponder (tag). This mode allows NFC devices to behave as a contactless card in application such as payments, ticketing and access control as shown on Figure 2.4

2.2.2 Security in Near Field Communication

This section gives an overview of security threats, possible attacks on NFC and ways to avoid them. As mentioned above, NFC is a short-range data exchange, which is used to make transactions, connect electronic devices with a touch etc. Even though the interaction distance is very short, the potential for attack is very high, which makes customers using NFC-enabled devices for transactions particularly wary. The best way to protect against possible attacks on NFC-enabled devices is to know the risks as elaborated in [Nea13, PCTRFSE13, ELS+13]. NFC-enabled devices operate in a short-range, the main question is how close a third party needs to be, to retrieve information during an attack [HB]. Attacking NFC-enabled devices depends also on the sender's operating mode.

2.2.2.1 Threats - Near Field Communication

The most common threats in NFC are as follows [NFC]:

Eavesdropping

The most common threat of NFC is known as eavesdropping. Two NFC devices communicate with each other with a purpose of transmitting various information between them. An attacker, also known as a third party, can intercept this communication. With the help of an antenna an attacker can receive or transmit signals. Whilst the third party is receiving or transmitting signals, different information from the third party can be eavesdropped. For instance within a communication between a smart phone and credit card reader, this may allow an attacker to have access to a person's credit card information.

Data Corruption and Modification

Another threat is data corruption or modification. An attacker intercepts the signal, modifies it. The third party is not interested in stealing information, the main point is to prevent the communication between the first party through sending invalid data or blocking the channel. Preventing this form of attack is very hard.

Data Insertion

Another security concern is data insertion. An attacker inserts invalid information between the first party who is in charge of data exchange between them. This type of risk is possible if the answering device needs a long time to respond and an attacker misuses this deficiency in the first party. After that, the answering device receives valid data from the sender and invalid data from the attacker which brings an overlap of the data and this data is corrupted [HB].

Man-in-the-Middle (MITM)-Attack

MITM interrupts the communication between the first party (Alice and Bob) and, therefore, is known as the third party or as an attacker (Eve) is shown on Figure 2.5. In this type of security concern, Eve successfully intercepts the signals between Alice and Bob. However, Eve also acts as a replier between Alice and Bob through data modification or at least having read or recorded them. This is one of the most complicated security risks in NFC, due to the short range communication. The simplest way to avoid the MITM is to use active-passive communication mode (see Section 2.2.1) [ELS+13].

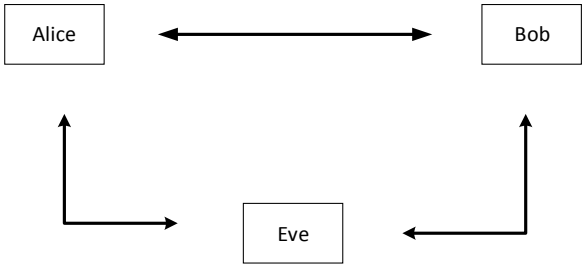


Figure 2.5: Man-in-the-Middle-Attack (adapted from [HB])

2.2.2.2 Prevention of Threats - Near Field Communication

As seen from Section 2.2.2.1 the probability of attack is very high. And this makes the customer or the first party always feel insecure whilst using NFC-enabled smart phones in their daily life. However, many security companies are working on it, to prevent an attacker from intercepting the signals between the first party. Mulliner et al. [Mul] performed the vulnerability analysis and attacks on NFC-enabled smart phones through the application of fuzzing. Their results take into account not only the NFC-subsystems but also software components. Further information on the prevention of threats in NFC are available under [HB, Nea13, PCTRFSE13, ELS+13].

2.3 Android

Android is an open source operating system based on Linux kernel [OHA13a]. It was found by Android, Inc in 2003 [Elg13] and later on in 2007 it was taken by the **Open Handset Alliance** (OHA) [OHA13b] which develops hardware and software for telecommunication. Android operating system is a platform designed for smart phones and tablet computers. Due to open source it allows the developer to modify the code for their own purposes and as a result the number of developers who develop applications (known as **apps**) is increasing dramatically.

2.3.1 Android System Architecture and Design

As mentioned at the beginning, Android is based on Linux kernel, its stack holds several layers, whereas each layer groups several programs [Tut13, Lin] as shown on Figure 2.6. Furthermore, Hardware Abstraction Layer (HAL) is used between hardware and the software stack, Android does not include all Linux utilities [Dev13].

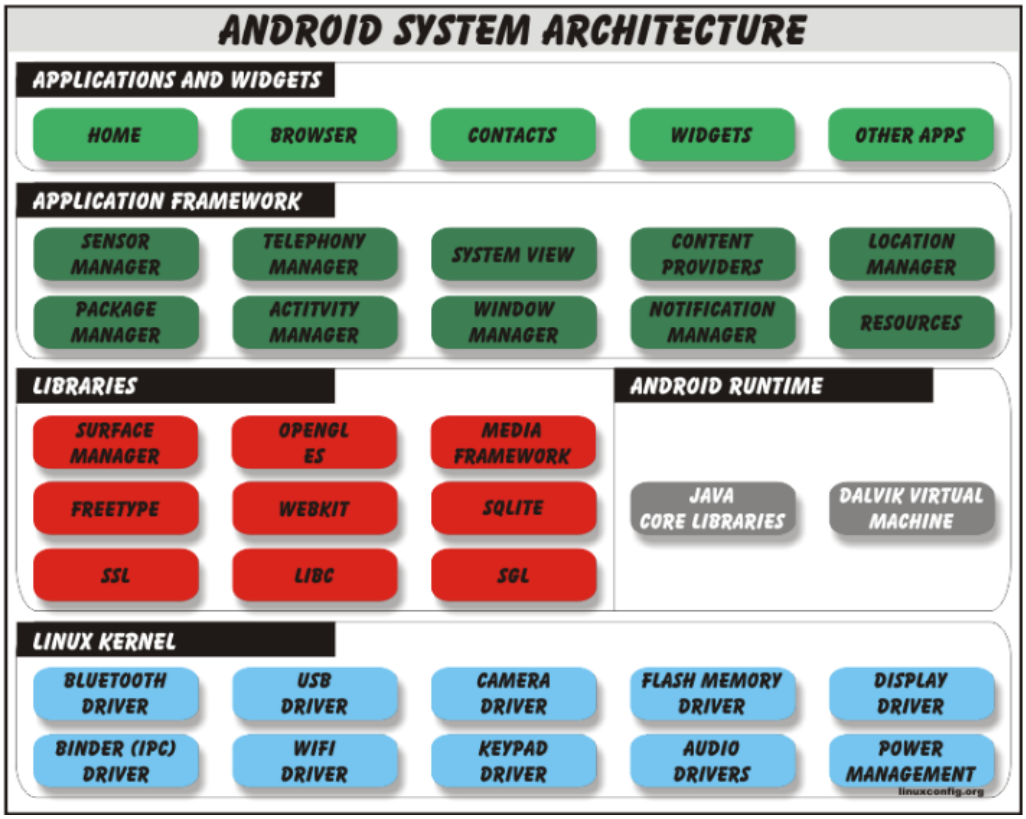


Figure 2.6: Android Architecture [Lin]

2.3.1.1 Linux Kernel

Linux kernel is placed in the bottom of Android stack and is the main point of the whole operating system. It provides basic system functionality such as memory management, process management, networking and other operating system services.

2.3.1.2 Libraries

Native libraries are placed on the top of Android stack, whereas many of them are preinstalled on a device. Native libraries such as 3D, 2D graphics, window manager etc. are written in C/C++ programming language.

2.3.1.3 Android Runtime

The third section of Android system architecture is known Android Runtime. This layer provides Dalvik Virtual Machine (DVM), which makes the use of Android feature using Java language. It is Java Virtual Machine, designed and optimized for Android. Android application developers can develope different applications written in Java programming language by using the core of libraries which are provided by Android runtime.

2.3.1.4 Application Framework

Android Application Framework in the form of Java classes provide basic high-level services. This framework allows the application developers to manage the basic Android functions such as resource management, voice call etc. in their applications.

2.3.1.5 Applications

It is placed at the topmost layer of the Android stack. It is a layer where the user installs their own applications such as Contact Books, Browser, Games etc.

2.4 Authentication Algorithms for Near Field Communication

One of the main problems in the cryptosystem is data eavesdroping while exchanging data between a receiver and a transmitter in embedded devices. Furthermore, since the famous attack against the German Enigma encryption [Ble] machine during World War II, the security of any product plays a crucial role.
As NFC-interface started to become very popular in smart phones, users/customers are using NFC-enabled smart phones for different purposes e.g. payment, accessing a building etc. While these and many other functionalities are provided by an NFC-enabled smart phone the possibility of attacking personal data is growing dramatically. Various ideas, projects have been presented in the past in order to avoid this negative approach.
This section gives an overview of the basics of cryptography which are necessary and used in this master's thesis. Further information on ECC is available at [HMV04, MvOV01, PP10, Riv].

As shown in Figure 2.7 *cryptography* and *cryptoanalysis* are not the main parts, they are only the branches of *cryptology* [DKS10].
Cryptography is the science of secure communication by hiding the original message from third parties (called attackers) and is divided into groups [Riv]:

- Symmetric Ciphers
- Asymmetric Ciphers
- Protocols

Cryptoanalysis is the science of breaking the hidden messages [ASb].

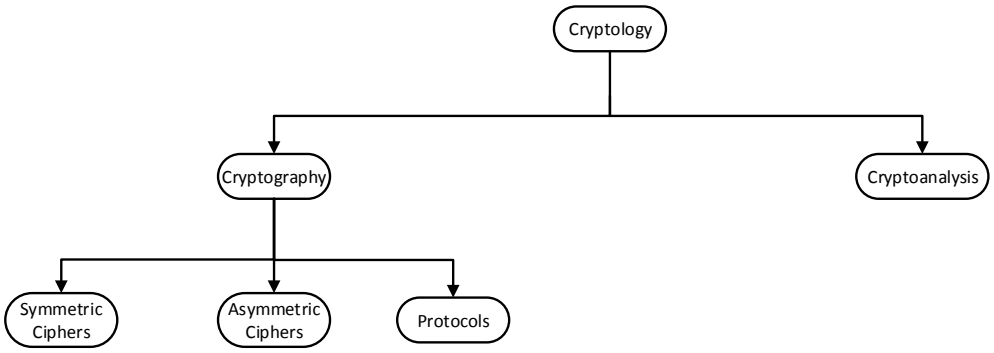


Figure 2.7: Overview of Cryptology (adapted from [PP10])

2.4.1 Symmetric Cryptography

Symmetric cryptography is also known as a symmetric-key, secret-key and single-key algorithms. The first party (Alice and Bob) as shown in Figure 2.8 use the same secret key for encryption of plaintext and decryption of ciphertext over an insecure channel. Alice and Bob must use the same secret-key otherwise the encryption and decryption fails or the third party (Oscar) may decrypt this key and hack (steal, modify) the exchanged information. Due to the access from both sides (Alice and Bob), the secret-key is represented as a shared key which is one of the main drawbacks in symmetric cryptograhy. Symmetric cryptography is divided into two parts: Stream Ciphers (encrypts bit individually) and Block Ciphers (encrypts a whole block) [PP10, p. 30]. In general it is not expensive to produce a strong key and process it. Hence, implementation in hardware is very simple and does not require much experience [HKhF11]. Data Encryption Standard (DES) is known as the most famous block cipher for encryption of the whole block [aI77]. However, where a higher level of security is required, data can be encrypted more than once, for instance, three times and it is assigned to as 3DES, Triple-DES. Advanced Encryption Standard (AES) is the most widely used symmetric cipher for the encryption of data [aes01].

2.4.2 Asymmetric Cryptography

Asymmetric cryptography or *public-key cryptography* requires two various keys: private and public key. Furthermore, the plaintext is encrypted and the digital signature is verified using a public key. The decryption of a ciphertext or creation of a digital signature is realized using private key [MvOV01, p, 283]. In comparison to symmetric cryptography (Section 2.4.1), the public-key cryptography is slower. Therefore it is only used for encryption of small data items, for instance, credit card numbers, PIN, and for authentication while its keys are generated mathematically. It is difficult to determine the private key from the generated public key. The public key may be published it is not secret, whereas the private key should only be distributed between the first party (Alice and Bob). The most widely used asymmetric cryptography scheme is **RSA** referred to (Rivest-Schamir-Adleman) algorithm [PP10, p. 174]. One of the main drawbacks of RSA is the integer

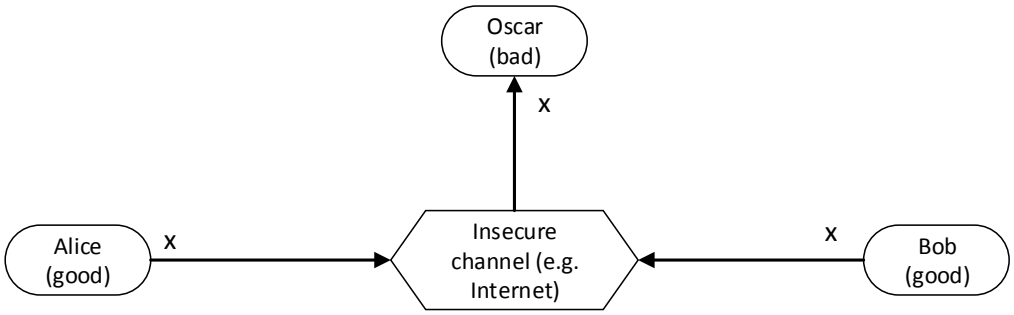


Figure 2.8: Symmetric Cryptography (adapted from [PP10])

factorization which is said to be a one-way function, whereas multiplication of two large prime numbers is very simple [RSS, AGGB11]. The main reason for an asymmetric cryptography is: for the exchange of symmetric keys. Further information on asymmetric cryptography is available in books, such as [MvOV01, Chapter 8.],[PP10, Chapter 7]

2.4.2.1 Diffie-Hellman Key Exchange

Whitfield Diffie and Martin Hellman proposed the first asymmetric scheme called *Diffie-Hellman key exchange (DH)* in 1976 [DH76]. However, there were also some other contributors who showed an interest in public-key exchange, for instance, Ralph Merkle[3], who was a major contributor to a public-key and as a gratitude Hellman proposed a new asymmetric scheme's name **Diffie-Hellman-Merkle Key Exchange** in 2002. With the help of Diffie-Hellman, two different parties establish a shared secret key and communicate over an insecure channel without knowledge of each other [Li10]. Although DH is a non authenticated protocol it provides some basics for authentication protocols and is used in Secure Shell (SSH) [MF06], Transport Layer Security (TLS) [DR08] and Internet Protocol Security (IPSec) [PP10, p. 206] [ZZ09]. The computations of DH are similar to RSA. The Diffie-Hellman key exchange protocol is shown on Figure 2.9 and works as follows:

DH's basic idea is the exponentiation in Z_p^* which is commutative, p prime, is a one-way function [PP10, p. 206]:

$$k = (\alpha^x)^y \equiv (\alpha^y)^x \bmod \ p \tag{2.1}$$

whereas k is a joint secret and can be further used as a shared session key between two parties. This joint secret key can be used also for symmetric algorithms such as AES or 3DES.

[3]http://www.merkle.com/

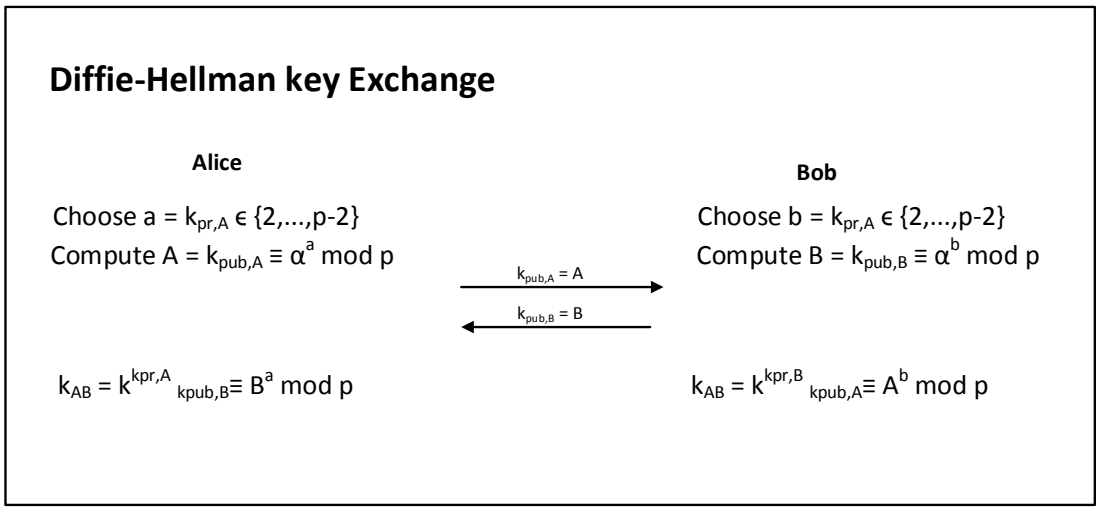


Figure 2.9: Difiie-Hellman Key Exchange (adapted from [PP10])

DH consists of two protocols: *set-up protocol* which consist of the following parameters:

- *Choose a large prime p.*
- *Choose an integer* $\alpha \in \{2, 3, ..., p-2\}$.
- *Publish p and* α

and *main protocol*.
As before, for the explanation of Diffie-Hellman key exchange protocol, Bob and Alice are taken into account as two different parties who communicate with each other over an insecure channel. Prime p and α are also known as domain parameters. As the first step Alice and Bob have to generate those domain parameters in the set-up phase and then generate the shared secret key as shown in Figure 2.9. As elaborated and shown in Figure 2.9 Alice and Bob generate the same session key $k_{AB} \equiv \alpha^{ab} \ mod \ p$, which is the main task of key-exchange protocol and helps to establish the communication over an insecure channel. Before generation of a shared secret key, Alice and Bob have to choose their private keys (a,b), and then compute the public keys (A,B) from randomly generated private keys. As mentioned in [Li10] DH also has its own drawbacks, for instance no information about the identities of the first part and also it is computationally intensive and it can not prevent the replay attack. Further information about Diffie-Hellman is available in book [PP10, p. 206]

2.4.2.2 Discrete Logarithm Problem (DLP)

In this master's thesis one of the aims is to implement an optimized ECC-based authentication protocol. The solution of DLP is not always simple, such that a cryptosystem depends on its hardness [oST92]. Public-key-infrastructure (PKI) system can be constructed with

the help of following algorithms such as Diffie-Hellman (see Section 2.4.2.1), Okamoto Conference-Key sharing scheme [Oka88], ElGamal Public-key cryptosystems [ElG85], all of which are based on DLP.
DLP is divided into two types[Zc09]:

- multiplicative group in finite field, for instance cyclic multiplicative group of the prime field and
- group of points on an elliptic curve over a finite field (see Section 2.4.3).

The following components are essential to understand DLP [PP10, p. 208] [Vad13]:

- *Group: is a set of elements G together with an operation* o *which combines two elements of G.*
- *Finite groups (G,* o*): is finite if it has a finite number of elements.*
- *Cyclic groups: a group G which contains an element* α *with maximum order* $ord(\alpha) = |G|$.
- *Subgroups:* (G, o) *is a cyclic group. Then every element of a* $a \in G$ *with* $ord(a) = \mathrm{s}$ *is the primitive element of a cyclic subgroup with* s *elements.*

Definition: *DLP in* Z_p^* *is given for a finite cyclic group* Z_p^* *of order p-1 and a primitive element* $\alpha \in Z_p^*$ *and another element* $\beta \in Z_p^*$. *The DLP is the problem of determining the integer* $\leq x \leq$ *p-1 such that:* $\alpha^x \equiv \beta \mod p$. *The integer x is called* ***Discrete Logarithm Problem (DLP)*** *of* β *to the base* α *as follows:* [PP10, p. 217].

$$x = \log_\alpha \beta \mod p \tag{2.2}$$

The pros of DLP in cryptography are that it is not restricted but can be defined over any cyclic groups. This is called Generalized Discrete Logarithm Problem (GDLP) [PP10, p. 218]. However, various numbers of algorithms for computing discrete logarithms exist. Some of them work only with cyclic group called Generic algorithms [Sho97] as follows:

- Brute-Force Search
- Shanks' Baby-Step-Giant-Step Method
- Pollard's Rho Method
- Pohling-Hellman Method

Hence, for those groups for which the DH problem can be solved, it is also possible to compute discrete logarithms efficiently [MW00]. As mentioned in [PP10, p. 216] cyclic groups can be used to solve them. A group of cryptography researchers [PO08] involved cyclic groups for the solution of DH and DLP.

2.4.3 Elliptic Curve Cryptosystems

Elliptic Curve Cryptography (ECC) is introduced as one of the newest members of public-key cryptography. As described in Sections 2.4.2.2, 2.4.2 RSA and discrete logarithm systems both provide a high security aspect with relative short operands. However, ECC in comparison to RSA and DH provides the same level of security but with the shorter operands; for instance shorter computations, shorter signatures and keys. ECC is based on generalized discrete logarithm problem. Table 2.3 shows the comparison of key sizes between ECC and RSA/Diffie-Hellman key exchange recommended by NIST[4] and also the key size protection used in encryption algorithm like DES and AES. Although to protect 128-bit AES keys, RSA or Diffie-Hellman requires 3072 bit parameters whereas ECC only 256 bits. Elliptic curve in cryptography was for the first time proposed by Neal Koblitz [Kob87] and Victor S.Miller [Mil86] in 1985. Even though it was suggested so early, the usage of elliptic curve in cryptography only started nearly 20 years later, in 2004 to 2005 and was approved by NIST in 2006.

Table 2.3: ECC and RSA/DH key size (adapted from [ASa])

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

2.4.3.1 ECC mathematical

Definition: *An elliptic curve over a field K is defined by an equation*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.3}$$

where a_1, a_2, a_3, a_4, $a_6 \in$ K and $\Delta \neq 0$, where Δ is the discriminant of E and is defined as follows:

$$\begin{aligned}
\Delta &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\
d_2 &= a_1^2 + 4a_2 \\
d_4 &= 2a_4 + a_1a_3 \\
d_6 &= a_3^2 + 4a_6 \\
d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2
\end{aligned}$$

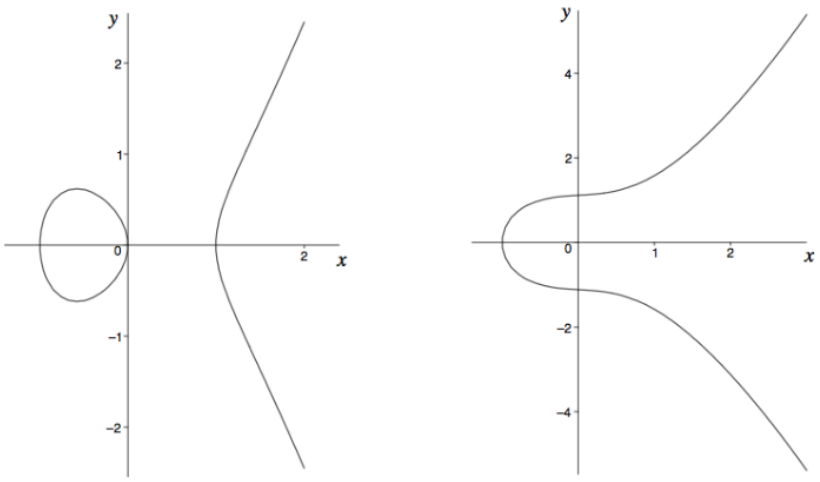
The Equation 2.3 for an elliptic curve over a field is called the Weierstrass equation [alo]. Where K is called underlying field and E/K means E is defined over field K.

There are different types of elliptic curves, each of them with their own specifications whereas they play a crucial role in security. Figure 2.10 shows two different types of curves

[4]National Institute of Standards and Technology

over the field R of real numbers.

If L is any extension field[5] *of K where E is defined over K, then the set of L-rational*



(a) $E_1 : y^2 = x^3 - x$ (b) $E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

Figure 2.10: Various elliptic curves over R [HMV04, p. 77]

point on E is as follows:

$$E(L) = \{(x, y) \in L \times L : y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0\} \cup \{\infty\}$$

2.4.3.2 An Algebraic Approach to Elliptic Curve

As a rule for adding two points in E(K) is taken *chord-and-tanget*. Let E be an elliptic curve over real numbers, whereas $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are two distinct points on curve. The geometric arithmetic computation is as follows [Gao93]:

Addition *of two distinct points on elliptic curve E(K), for instance, point P and Q which are not negative to each other* is shown in Figure 2.11 (a) and follows:
$R = P + Q, (x_R, y_R) = (x_P, y_P) + (x_Q, y_Q)$

$$s = \frac{(y_P - y_Q)}{(x_P - x_Q)}$$
$$x_R = s^2 - x_P - x_Q \text{ and } y_R = -y_P + s(x_P - x_R)$$

Whereas s is a slope through point P and Q, this line intersects the elliptic curve at a third point. Reflection on x-axis represents the third point R.

Doubling *of any point on elliptic curve E(K), for instance, point* $P = (x_P, y_P)$, *whereas* $P \neq -P$, *then* $2P = R$ as shown in Figure 2.11 (b) is as follows:

[5] $q = p^m$, if $m \geq 2$, then F is called an extension field

$$s = \frac{(3x_P^2 n \ + \ a)}{(y_P)}$$
$$x_R \ = \ s^2 \ - \ 2x_P \ and \ y_R \ = \ -y_P \ + \ s(x_P \ - \ x_R)$$

Whereas a is one of the parameters chosen from the elliptic curve and s is the tangent on the point P which intersects the elliptic curve at a second point. Reflection on a-axis is the third point R.

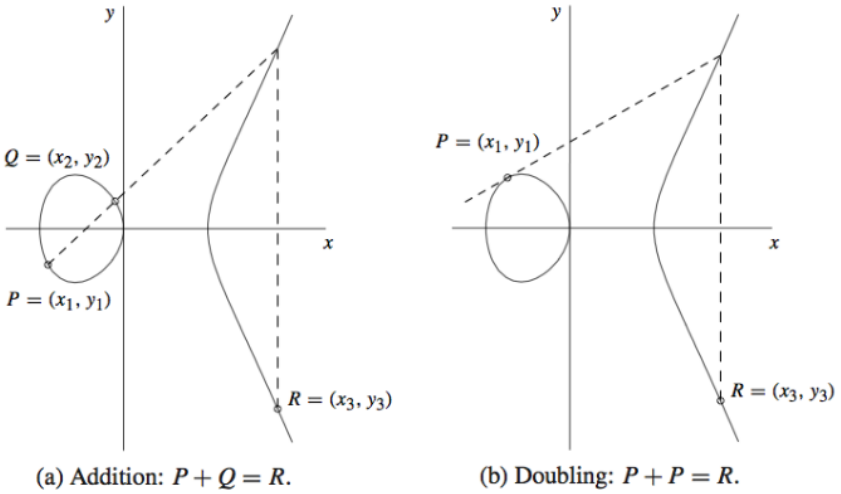


Figure 2.11: Point Addition and Doubling of elliptic curve [HMV04, p. 80]

2.4.3.3 Finite Fields

As seen from the above elliptic curve operations and various elliptic curves depicted in Figure 2.10 all of them are on real numbers. However, those operations are very slow for cryptography. In cryptography, operations need to be faster. Elliptic curve cryptography is defined over two finite fields [MS13]:

- Prime field F_p
- Binary field F_{2^m}

For the finite fields in the cryptography there are always necessary curves with finitely large number op points. The *order* of finite field is the number of elements in the field. Finite field F of order q for example $q = p^m$, where p is a prime number called characteristic of F and m is a positive integer, if m = 1 then finite field F is called a **prime field** (see Section 2.4.3.4).

2.4.3.4 Elliptic Curve over Prime Finite Fields F_p

Prime field F_p equation for the elliptic curve is $y^2 \mod (p) = x^3 + ax + b \mod (p)$, where p is a prime number and consists of positive integers {0, 1, 2,..., p-1} with all elliptic curve

operations such as addition, multiplication, subtraction, division, modulation and finite field of order p. Prime field F_p is called p modulus of F_p. Division operation in elliptic curve cryptography is called *reduction modulo*.

2.4.3.5 Group law for E/K : $y^2 = x^3 + ax + b$

Geometric description helps to derive the algebraic formulas for group law. The formulas below show addition and doubling of various points on elliptic curves E over prime field F_p where p > 3 is a prime of the simplified Weierstrass form in affine coordinates. Although the arithmetic operations as in real numbers will not work for prime field F_p, they have to be adapted as follows.

Definition: *Group law of elliptic curves over prime field* F_p
1. Identity. $P + \infty = \infty + P = P$ for all P $\in E(K)$

2. Negatives. If P=(x, y) $\in E(K)$, then $(x, y) + (x, -y) = \infty$. Therefore $-P = (x, -y)$

3. Point addition. Let $P = (x_1, y_1) \in E(K)$ and $Q = (x_2, y_2) \in E(K)$, where $P \neq \pm Q$, $P + Q = (x_3, y_3)$, where

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 \textit{ and } y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1.$$

4. Point doubling. Let $P = (x_1, y_1) \in E(K)$, where $P \neq -P$, Then 2P = (x_3, x_3), *where*

$$x_3 = \left(\frac{3x_1^2 + a}{2_{y1}}\right)^2 \textit{ and } y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1.$$

On the other side, information of elliptic curve defined over binary field F_{2^m} about the point addition, doubling etc. is available in [MS13], [HMV04, Chapter 2].

2.4.3.6 Elliptic Curve Domain Parameters over Prime Field F_p

Elliptic curve domain parameters over prime field F_p are [1.]:

$$T = (p, a, b, G, n, h)$$

whereas p specifies the finite field F_p, two elements $a, b \in F_p$ specifying an elliptic curve $E(F_p)$, $G = (x_G, y_G)$ is a base point on $E(F_p)$, a prime n is the order of G, and an integer h is the cofactor $h = \#E(F_p)$.
According to [fECS00], in the security aspect the elliptic curve domain parameters for finite field such as binary F_{2^m} and primary field F_p differ from each other and are used by various ECC standards such as ANSI X9.62 [X9.13a], ANSI X9.63 [X9.13b], and IEEE P1363 [P1300]. ECC domain parameters over F_p use a special form of primes for their field order p, which facilitates an especially efficient implementation. As mentioned in the

beginning ECC plays a crucial role in security level, therefore, ECC domain parameters over F_p consist of two different types of parameters: Koblitz curve [HMV04, p. 114] and random chosen parameters.

Further information on recommended elliptic curve domain parameters defined over F_p and F_{2^m} is available at [fECS00].

2.4.3.7 Affine Coordinates

Let's assume a method kP, where k is an integer and P is a point on elliptic curve E defined over a field F_q. This is called a point multiplication or scalar multiplication, it is used for every purpose in cryptography, for example, for encryption, decryption, signature, key exchange etc. Elliptic curve points can be further represented as a pair of integers in a finite field known as the affine coordinate representation. In cryptography the arithmetic operations have to be fast and need fewer resources. There is another method for point multiplication in elliptic curve called Montgomery multiplication which was introduced for the first time in 1985 by Peter Montgomery [Mon85].

Further information about affine coordinates is available at [HMV04, Chapter 3.3].

2.4.3.8 Projective Coordinates

Affine coordinates are used in the elliptic curve operations such as addition, doubling etc. Each of them required several field multiplications and a field of inversion. In comparison to multiplication, inversion in Field K is significantly expensive. Therefore, it is more advantageous to represent points using projective coordinates. In projective coordinates a new point Z is inserted $(X : Y : Z) = \{(\lambda^c X, \lambda^d Y,) : \lambda \in K^*\}$ and is called projective point, where c and d are positive integers [HMV04]. Elliptic curve equation for prime field using standard projective coordinates is as follows:

$$Y^2Z = X^3 + AXZ^2 + BZ^3 \tag{2.4}$$

An affine point (x_1, y_1) becomes a projective point $(x_1, y_1, 1)$ [Cas].

2.4.3.9 Montgomery Multiplication

This section shows only the basics of Montgomery multiplication, for more details please see [HMV04, Chapter. 2].
Montgomery multiplication was introduced in 1985 by Peter Montgomery [Mon85, Chapter. 2], and replaces division in classical reduction algorithms with less expensive operations. Montgomery multiplication is not efficient for a single modular multiplication, although it is used in computations such as modular exponentiation where many multiplications are performed (several hundred bits). One of the most expensive arithmetic operations in cryptography is multiplication, whereas Montgomery multiplication is replacing this critical problem with less inversion and is used by various cryptosystems: RSA and DSA which are based on arithmetic operations (multiplications, modulation). Another reduction method called *Barret reduction* which is investigated in 1986 by Barret [Bar86] uses the following computing

$$c = a \mod p$$

and can be considered as a fast division algorithm. Montgomery reduction also uses the same strategy to accelerate the scalar multiplication of elliptic curve points with the following equation

$$c = a \cdot b \mod p$$

Whereas the Montgomery multiplication procedure is as follows: Let $R > p$ with $gcd(R, p) = 1$, and for input $z < pR$, a Montgomery reduction produces $zR^{-1} \mod p$. If p is odd, then $R = 2^{W_t}$ may be selected and division by R is not expensive which is the main point of Montgomery reduction. If $p' = -p^{-1} \mod R$, then $c = zR^{-1} \mod p$ is a Montgomery multiplication which is obtained by the following method [HMV04, Section. 2.2.4]:

$$\begin{aligned} c &\leftarrow (z + (zp \mod R)p)/R, \\ &if\ c \geq p\ then\ c \leftarrow c - p \end{aligned}$$

with t(t+1) single-precision multiplications and no divisions.
Lets have two different inputs such as $x \in [0, p)$ and $y \in [0, p)$ where each of them must be transformed e.g. $\widetilde{x} = xR \mod p$. It has to be noted that $(x, y)R^{-1} \mod p = (xy)R \mod p$ can be also calculated using the Montgomery reduction method. Although Montgomery multiplication of two inputs is defined as follows [HMV04, Section. 2.2.4]:

$$Mont(\widetilde{x}, \widetilde{y}) = \widetilde{x}\widetilde{y}R^{-1} \mod p = xyR \mod p \tag{2.5}$$

Further information on the benefits of Montgomery multiplication on RFID is available at [LWaYX08, BDD].

2.4.3.10 Elliptic Curve Diffie-Hellman Key Exchange Protocol

The key exchange method between two parties in cryptosystems became one of the most useful methods of ensuring the data exchange between those parties. Elliptic Curve Diffie-Hellman Key Exchange's (ECDH) (see Section 2.4.2.1) the first condition is the agreement on domain parameter on which the elliptic curve is suitable [PP10, p. 250].

ECDH Domain Paramters

- 1. *Choose a prime p and the elliptic curve*

$$E : y^2 \equiv x^3 + a + b \mod p$$

- 2. *Choose a primitive element* $P = (x_P, y_P)$

  Domain parameters consist of the curve coefficients (a,b), the prime p and the primitive elements P.
  Further information on ECC domain parameters over Prime field F_P see Section 2.4.3.6 and book [PP10, Chapter. 2]

More about the functionality of ECDH is explained in Section 2.4.2.1.

2.4.3.11 The Elliptic Curve Digital Signature Algorithm (ECDSA)

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a variant of Digital Signature Algorithm which uses elliptic curve cryptography (see Section 2.4.3). It performs numerous advantages over RSA and DL-schemes like Elgamal. It uses shorter bit lengths than RSA and DL, this affects a shorter processing time, signature generation and verification. Due to those and many other advantages of ECDSA in 1998, this algorithm was standardized in the USA by American National Standards Institute (ANSI). For more details on ECDSA algorithm, please follow [PP10, Chapter 10].

Key Generation

The generation of ECDSA public key and private key is as follows:

1. *Select an elliptic curve E(a,b),*
2. *Choose a random integer d in interval [1,n-1],*
3. *Compute $B = d \cdot A$, where A is a point on elliptic curve,*
4. *Return public key B, and private key d*

Bit length of n must be at least 160 bit or more for higher security.

Signature Generation

ECDSA consists of a pair of integers **r,s** of the signed message. Signature of message x is computed by using the public and private key as follows:

1. *Choose an integer as random ephemeral key k_E in interval* $[1, n-1]$
2. *Compute $R = k_E A$,*
3. *Let $r = x_R$, x_R : x-coordinate of point R*
4. *Compute* $s \equiv (h(x) + d \cdot r)k_E^{-1} \mod (n)$

Although, for computation of signature s in step 3, the message x has to be hashed by using the function h.

Signature Verification

The generated signature shown above, has to be verified by the receiver as follows:

1. *Ensure that s is in interval* $[1, n-1]$,
2. *Compute auxiliary value* $w \equiv s^{-1} \mod n$,
3. *Compute auxiliary value* $u_1 \equiv w \cdot h(x) \mod n$,
4. *Compute auxiliary value* $u_2 \equiv w \cdot r \mod n$,

5. *Compute* $P = u_1A + u_2B$,
6. *The verification* $ver_{k_{pub}}(x,(r,s))$, *where* k_{pub} *is a public key*

$$x_P = \begin{cases} \equiv r \mod n \implies \text{valid signature} \\ \not\equiv r \mod n \implies \text{invalid signature} \end{cases}$$

The last step from the signature verification process x_P notifies the x coordinate of point P. If the point x_P has the same value as the signature parameter r *modulo* n, it is a valid signature (r, s) otherwise, the accepted signature from the receiver is invalid.

2.4.4 ECC in this Master Thesis

The basics of cryptography, particularly DH-key exchange which has been elaborated above, helps to construct and implement in this master's thesis an optimized ECC-based authentication protocol between an Android-NFC enabled smart phone and a security controller used as an RFID tag. The Android platform already supports ECC, which gives more reasons for integration of this cryptographic method for improving the authentication protocol. By showing the advantages of using ECC in authentication protocol, we decided also to use the DH-key exchange method. Furthermore, ECC with a shorter bit length offers a faster data processing time, requires less energy and provides a higher security level compared to RSA, etc.

2.5 Similar Projects

This section gives an overview of various projects where ECC is used in the purpose of an authentication protocol. Furthermore, different projects based on NFC-enabled devices are shown, with the purpose of ECC integration. Many researchers showed the advantages of using ECC in the security aspect, for instance, less power, energy efficiency, short operation time and short bit-length.

2.5.1 ECC Implementation

One of the most expensive and the most complicated operations in cryptographic schemes based on ECC for smart phones is the scalar multiplication. Various algorithms have been presented by different authors on this principle. Each of them presented their improvements; on power efficiency, time duration etc. Reyes et al. [RCMSDP13] treated five different methods for evaluation of key generation on a single scalar multiplication over prime and binary fields including very sophisticated methods. A P500h LG smart phone is chosen as a test platform which includes an ARM processor at 600 Mhz and Android 2.2 as operating system. They came to a result as shown in Figure 2.12 that scalar multiplication for ECC defined over prime field is 8 times faster and the best method performer was

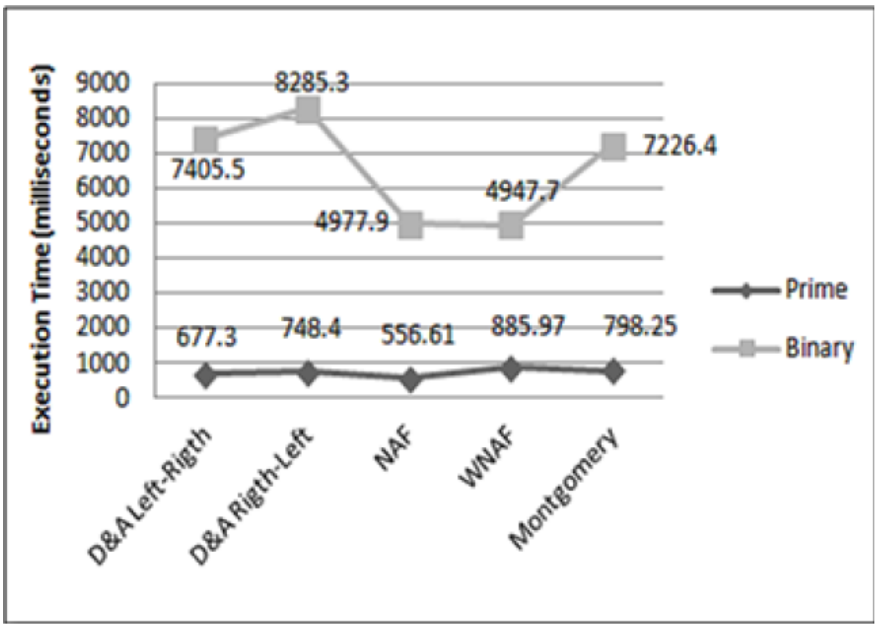


Figure 2.12: Overview of Scalar Multiplication Execution Time over Prime and Binary Fields [RCMSDP13]

NAF method. Although, for binary field the best method performer was window Non-Adjacent Form (wNAF). Furthermore, Reyes during their implementation also evaluated the execution time and the memory usage for a scalar multiplication.

As shown in Figure 2.12 the software implementation of ECC over binary field is slower than primary field, Shi et al [SY08] showed that implementation over binary field is still slow on low-end processors, especially on embedded devices such as sensor nodes. As a main parameter which affects on slowing the ECC implementation over binary field, Shi et al. suggested to be a word size. They used as a test platform for their methods a Pentium 4 including a processor at $3\,GHz$, and running Debian Linux system.
Further information on software implementation of NIST-recommended elliptic curves over binary field is available at [TFHA+11] [HHM00].

2.5.1.1 Software implementation of NIST Curves over Prime Field

Implementation of NIST curves over finite fields for embedded systems is one of the main topics nowadays. Various assumptions and results were elaborated by many authors. Whereas, many of these results assume that implementation of NIST Curves over prime field is faster in comparison to binary field [BHLM01]. Furthermore, the implementation of NIST Curves over prime field in comparison to binary field is cheaper.

Table 2.4 shows timing results for operations of the NIST prime fields simulated on a Pentium II 400 MHz workstation. Brown et al. [BHLM01] showed that primary field F_{p192}

[99] Coded primarily in C
[100] Uses 32×32 multiply-and-add
[101] Uses 32×32 multiply

Table 2.4: Comparison of NIST prime fields operations in μs (adapted from [BHLM01])

Field Operations	F_{p192} [99]	F_{p192}	F_{p224}	F_{p256}	F_{p384}	F_{p521}
Addition (Algorithm 1)	0.235	0.097	0.114	0.123	0.169	0.162
Substraction ([Algorithm 2)	0.234	0.094	0.112	0.125	0.158	0.15
Modular reduction						
Barret reduction (Algorithm 6)	3.645	1.021	1.462	1.543	3.004	5.448
Fast reduction (Algorithm 7-11)	0.223	0.203	0.261	0.522	0.728	0.503
Multiplication (including fast reduction)						
Classical (Algorithm 3)	1.238 [100]	0.823	1.074	1.568	2.884	4.771
Karatsuba	2.654[101]	1.758	2.347	2.844	-	-
Squaring (including fast reduction)						
Classical (Algorithm 4)	-	0.705	0.913	1.358	2.438	3.864
Algorithm 5	1.951 [101]	1.005	1.284	1.867	3.409	5.628
Inversion (Algorithm 12)	146.21	66.30	88.26	115.90	249.69	423.21

in the first column is written in C without the aid of hand-coded assembly code[6] whereas the other NIST curves over primary field are written in assembly and show a better timings result. In the case of hand-coded examples the NIST curves show a better performance for classical multiplication due to easy and limited insertion of assembly code. A Microsoft C (professional edition) compiler was chosen, whereas for assembler a "Netwide Assembler" NASM and for point multiplication (kP) a NAF method is used. Brown et al. suggested the usage of Chudnovsky over affine in Window NAF for large inverse multiplication ratio. They also achieved by using a simpler NAF method with Jacobian coordinates a very fast point multiplication with a less code.

For a fast addition and doubling of points in ECC there is no such system available. Even though Jacobian coordinates do faster doublings but slower additions than Chudnovsky Jacobian Coordinates. Cohen et al. [CMO98] proposed a new strategy for ECC exponentiation by using mixed coordinates. They used the best possible system for doublings and additions. They also analyzed timings for the modified Jacobian coordinates; whereas the new coordinates showed an improvement on computation time to approximately 1708.2 M[7] of 160-bit elliptic curve exponentiation. In general the new strategy presented by Cohen et al. reduces the computation time by more then 14 %.
Qian et al. [DRR13] treated a timing analysis in a software and hardware for NIST elliptic curves over prime fields. However, the software implementation for statistical analysis of the timing results of modular multiplication on NIST prime curve P-192 with two 160-bit random inputs achieves 1.47 μs and 1.96 μs.

2.5.1.2 Authentication Protocol using ECC

In general authentication is one of the main problems for wireless communication. Due to various attacks a lot of information is intercepted these days. Authors of different publications showed their results for an authentication protocol achieved in different manners, for instance, by using ECC. Bock et al. [BBD$^+$] proposes an asymmetric authentication protocol based on ECC. However, in case of preventing tag cloning this is mainly used

[6] 32×32 multiply with (add) is used in assembly. Standard C does not support 32×32 multiply and has no direct access to the carry bit

[7] Multiplication

for RFID (see Section 2.1) products. Whereas, the challenge-response is calculated by multiplication of an elliptic curve point with a random tag's secret key. Multiplication is done via Montgomery multiplication algorithm. In fact this Montgomery multiplication of an asymmetric authentication is used for curves defined over binary field, although in this master's thesis the proposed authentication protocol is based on this idea, but for elliptic curve defined over prime field. The ISO/IEC 15693/ 18000-3 Mode 1 compliant RFID tag is used which is a cooperation product between Infineon[8] and Siemens Corporate Technology[9]. During ECC calculation the energy spent achieved $10\,\mu J$ and the overall tag's size is $0.8\,mm^2$ in a 220 nm technology.

The main problem, as mentioned above, in wireless communication is data eavesdropping, therefore, Zhang et al. [JF09] proposed an efficient authentication and key agreement protocol based on ECC. However, for a key agreement protocol Diffie-Hellman key exchange, modular concept, a hash function is used. Their results show that this protocol for shorter keys provides a very high security concept and also reduces the bandwidth. On the other side, processing of an authentication protocol requires a high power, where Hutter et al. [HFP10] presented a 192-bit ECDSA processor for RFID authentication. The proposed processor has a chip size of totally 19 115 gates equivalents and signs a message within 859 188 clock cycles (127 ms at 6.78 MHz).

Further projects on authentication protocol for RFID using ECC are available at [HJWH11, HLL12, ARH08, Hut, Fel04, PWmMlZsL11]

2.5.2 Attacks on Authentication Protocol

Even though the distance of interaction for data exchange of NFC devices is very short, only up to a few cm, this seems to be again a serious problem for security experts. Lowe et al. [Low95] presented an attack on the Needhman-Schroeder public-key authentication protocol. The aim of this attack is to allow the mimic of an attacker as another agent. Lowe et al. described an attack on the Needham-Schroeder public-key protocol, and finally he discovered that it is as easy to change the protocol as it is to prevent an attack on the public-key authentication.

However, an authentication protocol consists of encryption and decryption of data and therefore Clark et al. [Cla96] analyzed the failing of these protocols to meet their goals. Clark et al. focused on analysis of the following attacks:

- **Freshness Attacks** *occur when an attacker from the previous run of a protocol recorded the message and replies in the current run of a protocol as a message.*
- **Type Flaws** *occur if a receiver accepts an original message as delivered from a sender. The received message is modified through different interpretation on the bit sequence.*
- **Parallel Session Attacks** *occurs when there is currently an execution of two or more protocols running. The messages of one protocol are used to form messages in another protocol.*

[8] http://www.infineon.com/cms/en/product/index.html
[9] http://www.siemens.com/entry/de/de/

- **Implementation Dependent Attacks** *occur when there is an interaction between a specific protocol and the actual encryption method.*
- **Binding Attacks** *are a binding between a public key and the corresponding agent.*

The NFC feature is integrated into various embedded devices especially on smart phones. Therefore, the risk of using NFC embedded devices especially with smart phones in daily life for different purposes such as for payment, accessing etc. is still present and unwanted even though many security experts are working hard on this topic. Verdult et al. [VK11] showed that installation of various viruses on NFC-enabled mobile devices became due to invocation of Bluetooth connection by NFC feature without user consent. Verdult et al. analysed that browsing to a passive object by NFC-enabled mobile devices leads to the **Content Sharing** feature. Only the passive objects (dataset) can be shared from an NFC-enabled device and not the installed applications.

NFC-enabled smart phones are used for different purposes but many of the users use NFC-enabled devices instead of wallet e.g. credit cards, access cards etc. The first NFC-enabled application for paying was launched by Google called **Google Wallet** [War13], which made a great impression on customers. Google wallet was later broken, which had negative implications on the user applications. Roland et al. [RLS13] evaluated a feasibility of the software-based relay attacks on Google Wallet. Roland et al. tested several workarounds and they were able to mount the software-based relay attacks. And finally, they also proposed a solution of avoiding software-based relay attacks on Google Wallet.
Further projects on attacks to an RFID mutual authentication protocol and to an NFC-enabled smart phone are available at [PHF+13, AAE+13, EMRY13, EA10, AGC10]

2.5.3 Design and Implementation of an NFC Interface for Home Appliances and Consumer Electronics

Bashagic et al. [Bas13] proposed an interaction between an NFC-enabled end-user and a passive RFID tag attached to non-wireless equipment. In addition, as depicted in Figure 2.13 an NFC Interface (NFC-I) target is designed and implemented, whereas the low-cost, energy efficiency and time duration are analyzed. Although for this interaction, sending

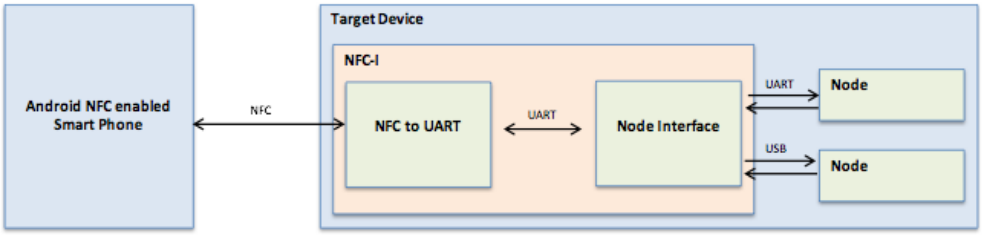


Figure 2.13: Interaction between Android NFC-enabled and NFC-I Target Devices [Bas13]

data to an NFCI-target device an Application Protocol Data Unit (APDU) is used, which consists of two types: Command Application Protocol Data Unit (C-APDU - request from NFC-enabled Smartphone to target device) and Response Application Protocol Data Unit (R-APDU - response from target device to NFC-enabled smart phone). For the data flow

within the whole system Druml et al.[DMB+12] implemented the basic communication libraries which are further used in this master's thesis.

2.5.4 Elliptic-Curve Cryptography on a Lightweight Microprocessor for RFID

Höller et al. [Höl13] tried to answer the common question regarding the security aspect: "Is RFID ready for software-based ECC?" However, not only software but also hardware is designed, implemented and evaluated as depicted in Figure 2.14. Montgomery ladder 1 for binary field defined over ECC, computes the projective representation X_2/Z_2 of the x-coordinate of the results $k \cdot P$.

Algorithm 1 Montgomery's method for scalar multiplication. Adapted from [BBD+, BHM08]

INPUT: $k = (k_l, \cdots, k_1)_2, x_P$ affine x-coordinate $of\ P$
OUTPUT: (X_1, Z_1) projective representation of the x-coordinate $of\ k \cdot P$
pick random value r
$X_1 \leftarrow r, Z_1 \leftarrow 0, X_2 \leftarrow rx_p, Z_2 \leftarrow r$
for $i \leftarrow l$ *downto* $t-1$ **do**
    if $k_i = 1$ **then**
        $T \leftarrow Z_1, Z_1 \leftarrow (X_1Z_2 + X_2Z_1)^2$
        $X_1 \leftarrow x_PZ_1 + X_1X_2TZ_2, T \leftarrow X_2$
        $X_2 \leftarrow X_2^4 + bZ_2^4, Z_2 \leftarrow T^2Z_2^2$
    else
        $T \leftarrow Z_2, Z_2 \leftarrow (X_2Z_1 + X_1Z_2)^2$
        $X_2 \leftarrow x_PZ_2 + X_2X_1TZ_1, T \leftarrow X_1$
        $X_1 \leftarrow X_1^4 + bZ_1^4, Z_1 \leftarrow T^2Z_1^2$
    end if
end for
if $\Delta(\mathrm{X}_1, Z_1, X_2, Z_2, x_p) \neq 0$ **then**
    return error
else
    RETURN (X_1, Z_1)
end if

Höller used the Montgomery method for scalar multiplication 1 defined over binary field for 163-bit-NIST elliptic curve to check if the software-based ECC is ready for RFID. The usage of the Montgomery Ladder Algorithm 1 for hardware implementations produces an adequate performance due to the following reason [BHM08]: in comparison to non-window methods for scalar multiplication, Montgomery Ladder provides a competitive performance. This algorithm has a highly regular structure.

Although the implementation of software-based ECC also depends on hardware capability, the hardware that has been designed fulfills this necessary requirement. Software-based ECC is implemented for NIST curves over binary field. Höller showed that using coprocessor results in a better performance in terms of energy efficiency and area.

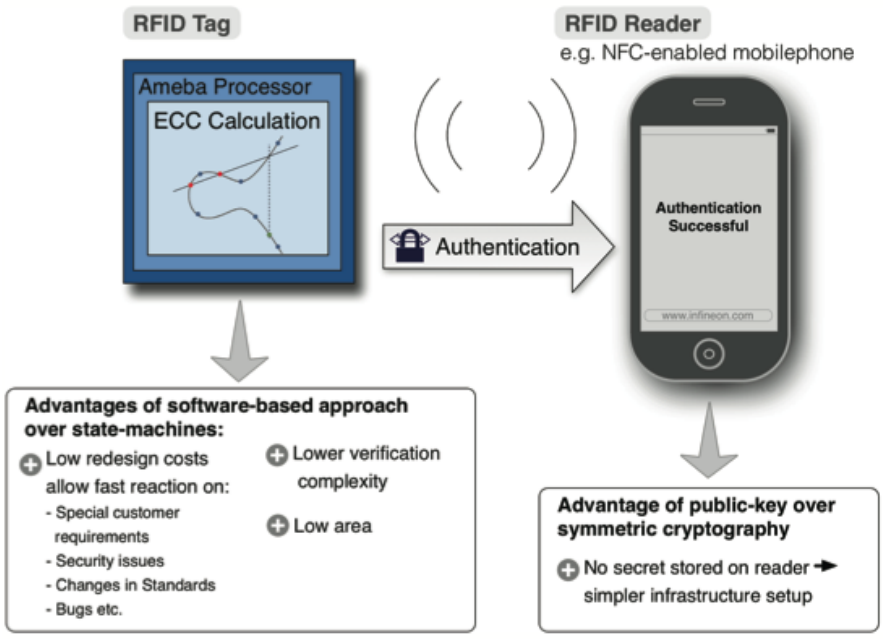


Figure 2.14: Elliptic-Curve Cryptography on a Lightweight Microprocessor for RFID [Höl13]

2.5.5 Trusted Device Interaction over NFC

Fioriello [TF13] presented another project which is a continuous work of Bashagic (see Section 2.5.3). He developed a secure channel communication between a smart phone and the last node (e.g. sensor). Furthermore, AES and ECC schemas are used, with special attention given to timing and energy consumption.
Figure 2.15 shows only one variant out of two different variants that Fioriello used for a connection establishment from source (e.g. NFC-enabled smart phone) to a target device over NFC-I. He also presented the energy consumption by using ECDH- and ECDSA-Algorithm for a secure communication. Although the interaction is started by the user (NFC-enabled smart phone) which sends its request (public key and signed public key) over NFC to NFC-I. As a result of using ECDH in securing the channel communication over WiFi, the energy consumption grows while also increasing key lengths. However, as a second variant of a connection establishment, he used an ECDH key exchange between smart phone and target device, where NFC is only a gateway between them. By using variant 2, Floriello got the following results: 160-bit curve needs the energy of around 63 % of a 224-bit curve, which is a difference of 27 %. In comparison to a 192-bit curve, the difference to 224-bit curve in terms of energy is about 12 %.

2.5.6 Various NFC Applications

NFC-enabled smart phones offer various services to customers. For example, they can be used as an access card, for the data exchange between NFC-enabled smart phones and for payment. Paying using NFC-enabled smart phones is one of the most common ways

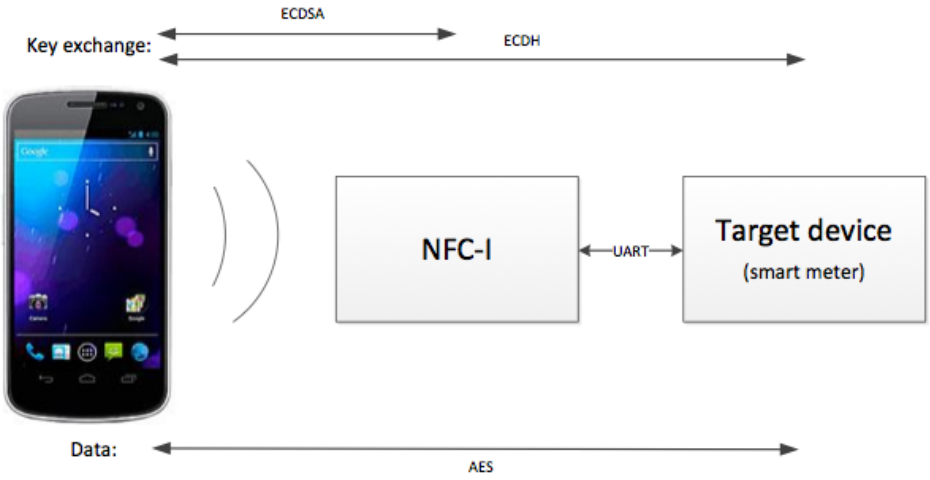


Figure 2.15: Overview on Trusted Device Interaction over NFC [TF13]

used by customers who replaced a wallet with various cards. As mentioned above, Google has already developed a very useful application for payment (Google Wallet [War13]), and there is a high percentage usage of this application in today's market. Furthermore, this allows users to store debit cards, credit cards, loyalty cards and some others on their mobile phone. Although, this application fulfills the requirement for payment; various authors have tested its security with various crypto algorithms.
Byunrae et al. [CK13] designed the NFC based Micro-payment system and security functions for the trade safety. The reason for designing such a system is due to the decreasing sales trend of small retailers and traditional markets. As a solution for this problem they proposed a very common way of using a Micro-payment model based on NFC, a tokenization technique to support the privacy of Micro-payment, and an MD authentication technique to attribute micro-payment as an aspect of IT. Tokenization is a very new and a rarely used methodology to secure specific data that is sensitive by replacing it with a non-sensitive and non-descript value set. MD authentication is a two way communication between a user and a server.

In order to avoid long queues to get onto a bus an NFC-enabled smart phone can be used for ticket payment. Various authors proposed different variants based on an NFC-enabled smart phone to make a ticket payment for any travel. However, a few of them tested the security of using a NFC-enabled smart phone on public transport. Hinterwälder et al. [HZB+13] proposed a new e-cash scheme based on an NFC-enabled smart phone which provides guarantees for security and user security; by using BlackBerry Bold 9900. One of the most common problems nowadays in embedded devices is their short battery life, due to the high need of power for the processing of a various number of threads. So Hinterwälder's et al. proposal is based on ECDHKeyAgreement which is available in BlackBerry API. They achieved a very good result in one aspect of processing time whereas a total execution time for spending a coin of all schemes did not exceed 400 ms, where the threshold in public transport domain takes around 300 ms. The point multiplication on BlackBerry needs around 4 ms, whereas on PC it takes 6 ms [HZB+13].

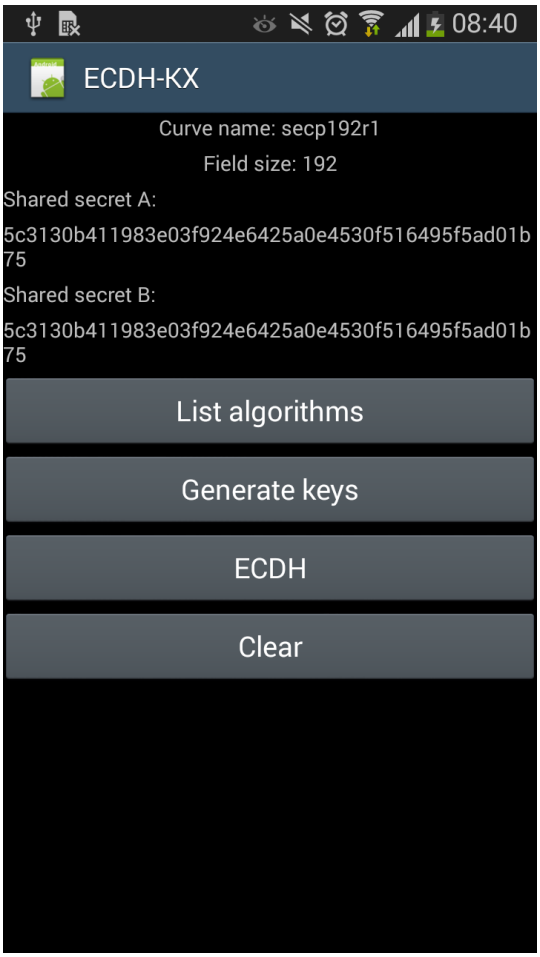


Figure 2.16: ECDH: Overview of shared secret key by using Spongy Castle [Ele13]

NFC-enabled smart phones are not only used for payment or data exchange but also in automotive systems. The eavesdropping and theft of data from car keys has recently become very common. Busold et al. [BTW+13] proposed an open security framework for secure smart phone-based immobilizers. However, their system is a generic security architecture which protects the electronic access to the smart phone. This system is implemented on the latest Android-based smart phone and a microSD smart card. Key agreement on an NFC-enabled smart phone plays a crucial role. The usage of ECDH in Android OS [Ele13] as shown in Figure 2.16 is only a simple application which shows the key agreement between sender and receiver (Bob and Alice). This master's thesis for the validation of an authentication request is based on the modular aspect of this simple application. Whereas, this application supports EC key generation, ECDH key exchange and ECDSA signatures. Bouncy Castle Library is thought to be the framework library for Android OS, but due to some conflict names between Android's OS class name and Bouncy Castle's class name another framework library as an alternative of Bouncy Castle is provided called Spongy Castle which is also further used for an implementation

of an authentication protocol in this master's thesis. This application shows only a simple performance of the functionality and usage of Spongy Castle framework library and key agreement in Android, it also uses standard NIST curves. A scalar multiplication of an elliptic point is also possible with the help of Spongy Castle framework library.

Chapter 3

Design

This chapter provides an overview of the system and design of this system. Furthermore, the main components used in this project for the system design are described in detail. Figure 3.1 shows an operational system overview for an authentication protocol. This system overview is used and applied in this master thesis for the design specifications and implementation of an optimized ECC-based authentication protocol between an NFC reader and an RFID tag. An Android NFC-enabled smart phone exchanges data over an NFC interface with a passive security controller used in this master thesis as an RFID tag. As elaborated in the Introductory Chapter 1, the main task of this project is to implement an optimized ECC-based authentication protocol for security controller-based applications which is applied between an Android NFC-enabled smart phone and the security controller.

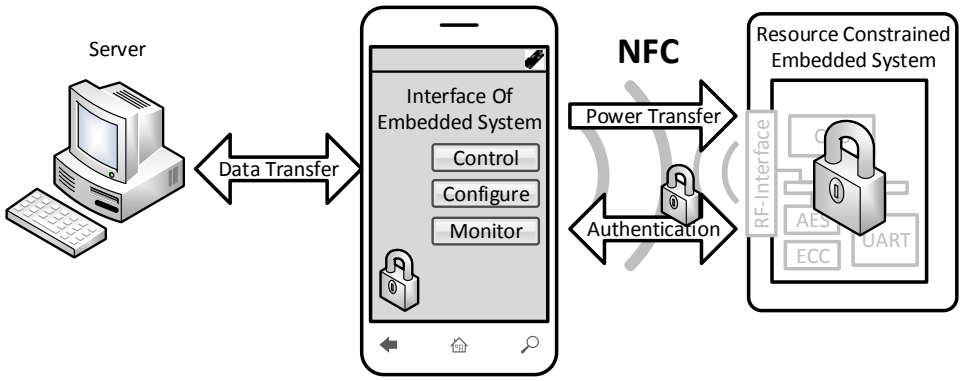


Figure 3.1: System Overview

As shown in Figure 3.2 the following components are described within this chapter:

- Smart Phone and Security Controller Interaction.
  Interaction for a purpose of data exchange between smart phone and security controller is initiated by an NFC-enabled smart phone. A simple graphical user interface

is designed and provided to the user for establishing this interaction.

- ECC Arithmetic operations for authentication protocol over Prime Field.
  This section gives an overview of the field arithmetic operations defined over prime field. An optimized ECC-based authentication protocol based on Montgomery Multiplication Algorithm is constructed for an NFC-reader and a security controller. Montgomery multiplication algorithm is realised and computed with the help of some basic field arithmetic operations e.g. Addition, Multiplication, Subtraction, Division.

- Optimized One-Way ECC Authentication Protocol.
  This section consists of the last four steps: Montgomery multiplication algorithm, software simulation, multiple APDUs, and single APDU interaction between an NFC-reader and the security controller. Finally, an optimized ECC-based authentication protocol is implemented, which is a challenge-response procedure based on an elliptic curve Diffie-Hellman key exchange. A high-level software simulation of this security controller authentication protocol is designed, whereby this simulation is used for further steps in an optimized ECC-based authentication protocol. Each of the Montgomery multiplication algorithm's field arithmetic operations is computed by sending a request to the security controller separately. On the other side, the security controller responds with a result for each of the NFC reader's requests e.g. Multiplication, Division, Addition etc. As a final version of an optimized ECC-based authentication protocol, a single request from the NFC reader is sent to the security controller, and a single response is returned from the security controller to the NFC reader.

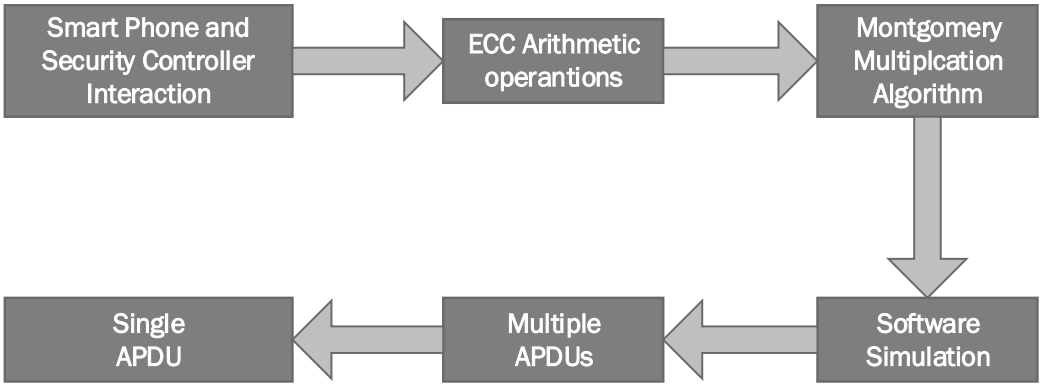


Figure 3.2: Design Flow of a Controller-based Brand Protection with Elliptic Curve Cryptography

Figure 3.3 shows an optimized ECC-based authentication program flow on an Android NFC-enabled smart phone. The flow of authentication protocol is as follows: after the software is started on an Android NFC-enabled smart phone, an initialization of user application occurs. The initialization process is provided by using *UserApplication* package which consists of two JAVA entities (*MainUI, and ECCAuthenticateUI*) as shown in Figure 3.4. With an initialization of a user application, it is assumed that the NFC-enabled

smart phone is ready to detect the security controller which is provided by the *MainUI* entity.

ECCAuthenticateUI is an intent which proves an optimized ECC-based authentication protocol in different ways e.g. computation of field arithmetic operations and calculation of Montgomery multiplication algorithm. As long as the user does not force the application to stop on an NFC-enabled smart phone, this application does not change to the next activity until a security controller is detected. Once a security controller is detected, a main screen is shown to the user, whereby some buttons are available for an ECC-based authentication protocol e.g. with multiple APDUs and single APDU. A special button is suitable only for a computation of a Montgomery multiplication algorithm on the security controller with the help of multiple APDUs. By clicking on any of the buttons dedicated for an authentication, an optimized ECC-based authentication protocol is carried out between an Android NFC-enabled smart phone and the security controller.

Authentication button sends a C-APDU from an Android NFC-enabled smart phone over

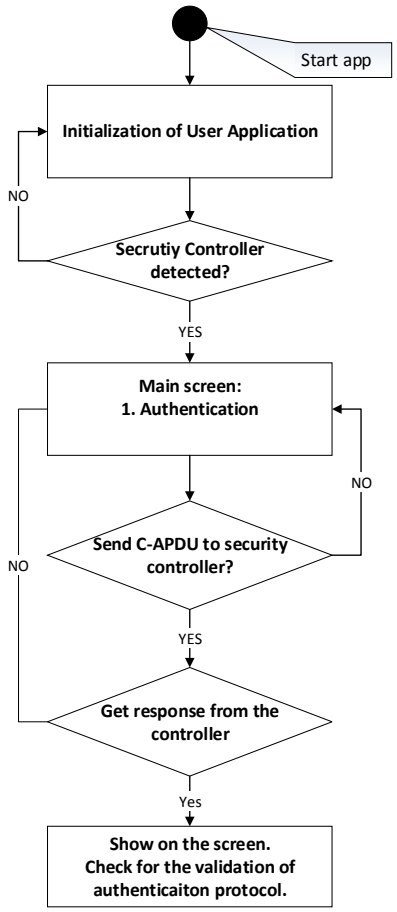


Figure 3.3: Authentication flow on NFC-enabled smart phone

an NFC interface to the security controller with a purpose of an authentication protocol. If a valid APDU message is sent from an Android NFC-enabled smart phone to the security controller, data is sent from the security controller to the NFC reader and is shown on the

screen. Hence, the validation of an optimized authentication protocol can be checked, by comparison of the expected data (simulated with software) with the responded data from the security controller. A valid authentication protocol is considered, if both of the pieces of data expected (simulated with software) and responded from the security controller are the same.

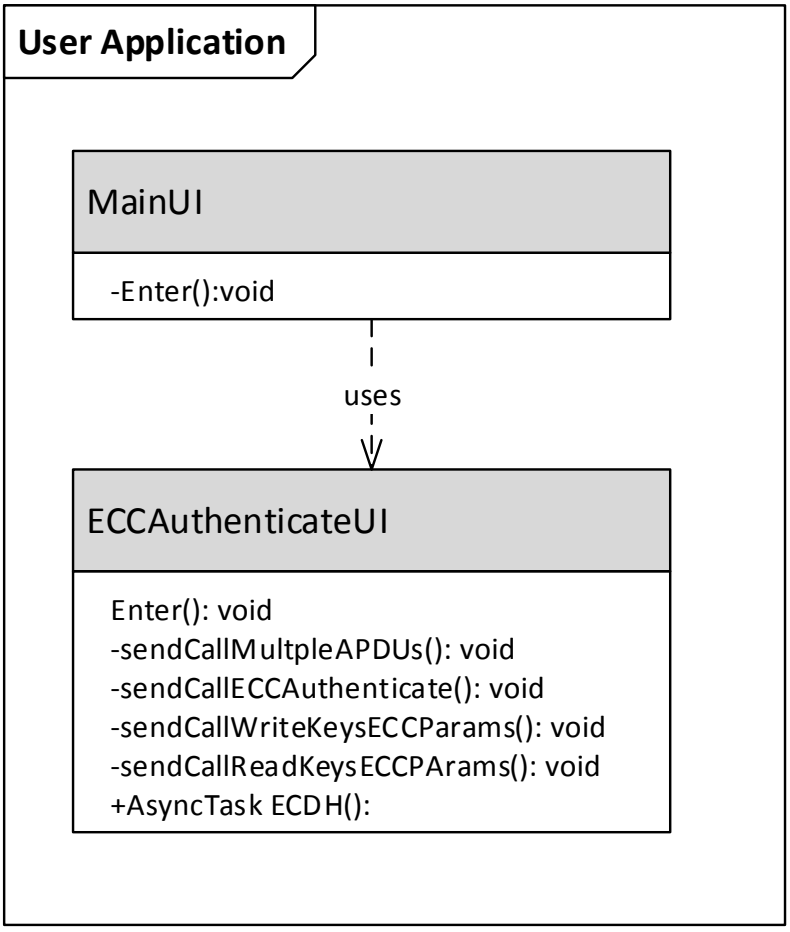


Figure 3.4: User Application Interface

3.1 Smart Phone and Security Controller Interaction

This section provides some information about an Android NFC-enabled smart phone and the security controller, which is used in this master's thesis for the purpose of an optimized ECC-based authentication protocol. Furthermore, the data exchange between an Android NFC-enabled smart phone and the security controller is described. A class diagram is depicted which shows the interaction between an Android NFC-enabled smart phone and the security controller. An NFC-enabled smart phone uses APDUs to store information and then sends the stored information on to the security controller over an NFC interface.

3.1.1 NFC-enabled Smart Phone

Initiation of an authentication protocol is taken by an Android NFC-enabled smart phone which forwards the data exchange with a security controller. As elaborated above, an NFC-enabled smart phone exchanges data with a passive security controller by using APDU in byte blocks over an NFC Interface. There are two categories of APDUs used for command and response, with their structure and characteristics being defined by ISO/IEC 7816-4 [78113]:

- Command Application Protocol Data Unit (C-APDU)
- Response Application Protocol Data Unit (R-APDU)

The structure and characteristics of C-APDU and R-APDU are shown in Table 3.1 and Table 3.2.

Table 3.1: Structure of Command APDU (adapted from [Rad02, p. 66])

Mandatory Header				**Optional Body**		
CLA	INS	P1	P2	Lc	Data Field	Le
Class byte	Instruction byte	Parameter byte 1	Parameter byte 2	Length of data byte	Data bytes	Expected length of the R-APDU

Table 3.2: Structure of Response APDU (adapted from [Rad02, p. 66])

Optional body	**Mandatory Trailer**	
Data Field	SW1	SW2
Data bytes	Status word 1	Status word 2

Android Manifest

AndroidManifest.xml is the necessary file for every Android application with exactly the same name in its root directory. Before running the application all the essential features (using internet, using NFC etc.) must be specified within this file.
These are some of the necessary components in this file:

- It names the JAVA package for the applications
- It gives a description of the application's components
- It specifies the name of the application
- It declares the permissions that an application needs to access the protected parts of the API
- It declares the permissions, required to interact with the application's components

3.1.2 Family of Infineon Security Controller

As elaborated above, for a secure data exchange an optimized ECC-based authentication protocol is applied between an Android NFC-enabled smart phone and the security controller. As shown in Figure 3.5 the security controller is provided by Infineon, and can be used as an access card (personal or government), payment solution, authentication, mobile communication etc.
The software part of Target of Evaluation (TOE) consists of the cryptographic libraries RSA and Elliptic Curves, the Toolbox and Base library. The EC library provides a high level interface to Elliptic Curve cryptography implemented on a hardware. The TOE consists of the following components: core system, memories, coprocessors, peripherals, security modules and analogue peripherals. Where Central Processing Unit (CPU), Memory Management Unit (MMU) and Memory Encryption/Decryption Unit (MED) are the main components of the core system. This Solid Flash family security controller is a secured smart card, therefore its coprocessor block contains the processors for RSA/EC and 3DES/AES processing, and a random number generator is a part of a peripheral block. Further information on the security controller family provided by Infenion is available at [AG, Hei]

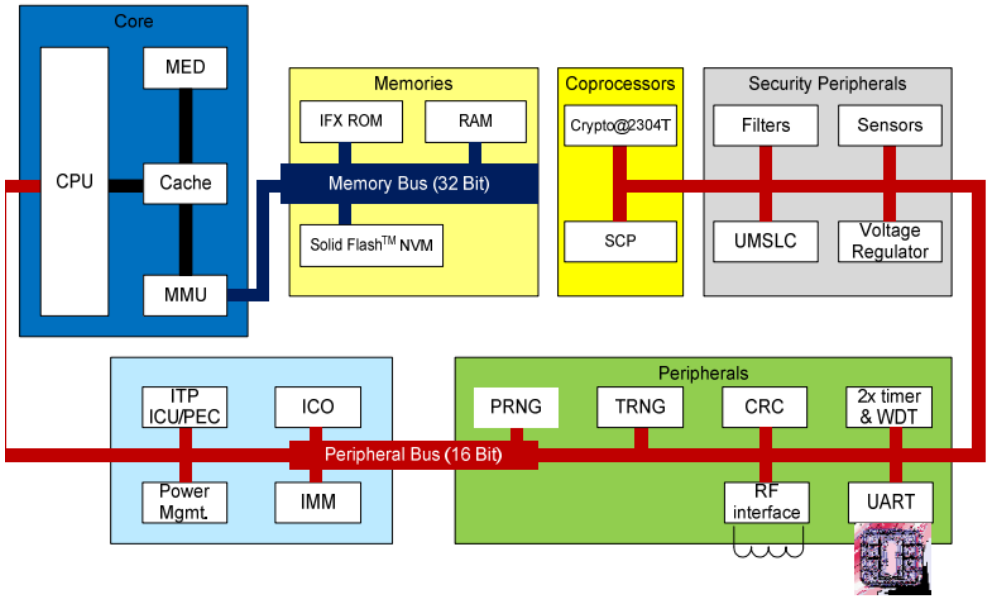


Figure 3.5: Example architecture of an Infineon Security Controller [Hei]

3.1.3 Interaction Data Flow

The software simulation and interaction deployment model between an Android NFC-enabled smart phone and Infineon security controller is shown in Figure 3.6. Interaction between an Android NFC-enabled smart phone and the security controller is initiated by a user. The user initiates this interaction with the help of various widgets provided by Android OS e.g. **TextView**, **Button** etc. In the case of interaction, different libraries

have to be used for the data exchange between an Android NFC-enabled smart phone and the security controller. In this scenario a *Crypto Library* is dedicated to a cryptography system, which provides all the necessary cryptographic libraries that are used to generate key pairs of a specified elliptic curve for an optimized ECC-based authentication protocol e.g. *PrivateKey, PublicKey, SharedKey*, etc.

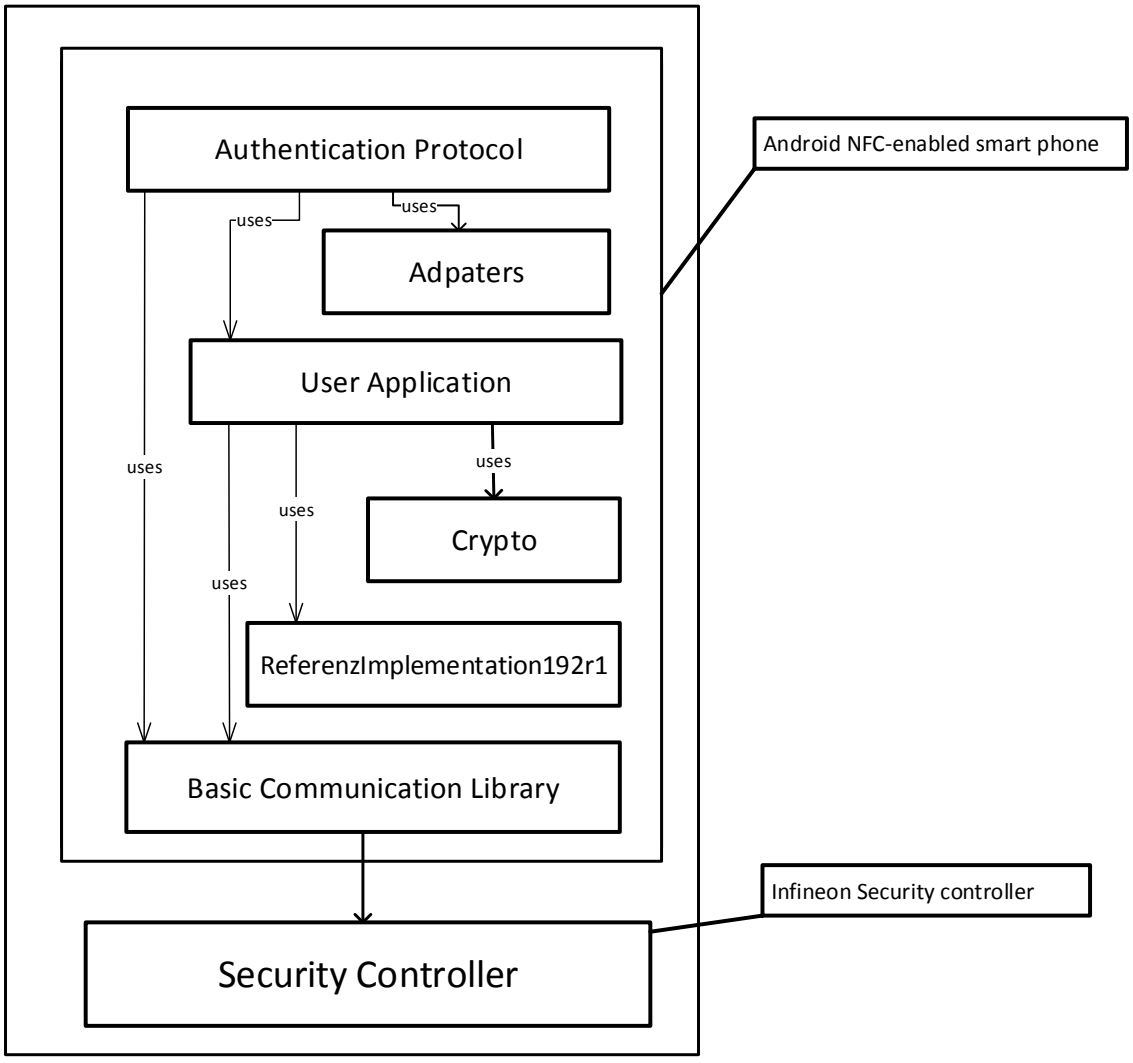


Figure 3.6: Deployment Model of Interaction Data Flow

Basic Communication Library provides a connection establishment between an Android NFC-enabled smart phone and the security controller. The user prepares messages, which are sent over an NFC interface to the security controller. These messages are prepared in byte blocks and with the help of C-APDU are sent as a request to the security controller. Depending on the request, a response (R-APDU) is returned from the security controller to the *Basic Communication Library* (NFC reader) which is further used for the specific purposes of an optimized ECC-based authentication protocol. The other libraries

used within Android NFC-enabled smart phone, will be explained in the further sections.

3.2 ECC Arithmetic Operations for Authentication Protocol Over Prime Field

According to various ideas, solutions, and results represented by various authors in Section 2.5, the authentication protocol of an RFID tag for distributive environments is mainly based on ECC defined over binary fields. However, within this master thesis, an authentication protocol is applied and implemented using optimized elliptic curves defined over prime field and is based on the work of Bock et al. There are various reasons for involving, integrating, and applying prime field for an optimized ECC-based authentication protocol.

For example the security controller used in this master's thesis supports SLE hardware multiplier therefore, its processing time for an authentication protocol in comparison to binary field is shorter and faster. This makes huge sense since less energy is needed for NFC-enabled smart phones to compute ECC defined over prime field, whereby the battery life is nowadays one of the most critical problems for embedded devices such as NFC-enabled smart phones.
The following arithmetic operations are adapted from the book *Guide to Elliptic Curve Cryptography* [HMV04], and are used in this project for an optimized ECC-based authentication protocol on NFC reader and an RFID tag.

Prime field arithmetic operation algorithms (addition and subtraction) are used within modular algorithms for addition and subtraction. The assignment of $(\epsilon, z) \leftarrow w$ is as follows [HMV04, p. 30]:

$$z \leftarrow w \mod 2^W, and$$
$$\epsilon \leftarrow 0 \; if \; w \in [0, 2^W\}, otherwise \; \epsilon \leftarrow 1.$$

where ϵ is called the *carry* bit for single-word addition, W stands for the bit architecture and is a multiple of 8.

3.2.1 Addition

The multi-precision addition algorithm 2 and multi-precision subtraction algorithm 3 are methods applied at a primary school, where the digits are added in the integers and track of carries are kept[GKPP06].

It is very important to mention that for both of the multi-precision algorithms (Addition 2) and (Subtraction 3) the basis b radix must be compatible with the underlying hardware. In this case a radix is chosen to be $b = 2$, which is compatible with a 32-bit architecture.

Algorithm 2 Multiprecision addition. Adapted from [HMV04]

INPUT: *Integers* $a, b \in [0, 2^{w_t})$
OUTPUT: (ϵ, c) *where* $c = a + b \mod 2^{W_t}$ *and* ϵ *is the carry bit*
$(\epsilon, C[0]) \leftarrow A[0] + B[0]$
for i *from* 1 *to* $t - 1$ **do**
    $(\epsilon, C[0]) \leftarrow A[i] + B[i] + \epsilon$
end for
RETURN (ϵ, c)

3.2.2 Subtraction

The working principle of multiprecision subtraction algorithm 3 is similar to multi-precision addition algorithm 2 but only with the following simple requirement. For multi-precision subtraction algorithm 3, it is often required that $a > b$. In order to avoid this, if the output c at the end of the computation has a value of -1, then one can perform the operation with the operands swapped [GKPP06]. The two's complement representation can be chosen by the developer.

Algorithm 3 Multiprecision subtraction. Adapted from [HMV04]

1: INPUT: *Integers* $a, b \in [0, 2^{w_t})$
2: OUTPUT: (ϵ, c) *where* $c = a - b \mod 2^{W_t}$ *and* ϵ *is the borrow*
3: $(\epsilon, C[0]) \leftarrow A[0] - B[0]$
4: **for** i *from* 1 *to* $t - 1$ **do**
5: $(\epsilon, C[0]) \leftarrow A[i] - B[i] - \epsilon$
6: **end for**
7: RETURN (ϵ, c)

3.2.3 Multiplication

Integer multiplication is considered to be one of the most critical arithmetic operations in the ECC. Point multiplication plays a crucial role in cryptography, whereas an integer multiplication in prime field F_p is accomplished by a simple multiplication between two operands, and then applying a reduction between them through a modulo operation.
As elaborated in Section 2.4.3, the elliptic curve point scalar multiplication can be achieved by performing the simplest method, the **Double-and-Add** method which is similar to the *square-and-multiply* method. Montgomery multiplication algorithm consists of various operations such as addition, multiplication, division, modulation and subtraction etc. Each of these operations (algorithms) should be manually implemented, if any operation is not provided by any programming language.

Algorithm 4 shows an integer multiplication using a basic operand scanning method. *Line 7* in Algorithm 4 performs an *inner product operation*, where each of the operands are a W-bit values.
If a CPU has a W-bit architecture for multiplication, then Algorithm 4 is used to get a result of 2W-bit multiplication [HMV04, p. 32].

Algorithm 4 Integer multiplication (operand scanning from). Adapted from [HMV04]

1: INPUT: $a,\ b \in [0, p-1]$
2: OUTPUT: $c = a \cdot b$
3: *Set* $C[i] \leftarrow 0$ *for* $0 \leq i \leq t-1$
4: **for** i *from* 0 *to* $t-1$ **do**
5: $\quad U \leftarrow 0$
6: $\quad$ **for** j *from* 0 *to* $j-1$ **do**
7: $\quad\quad (UV)[i+j] + A[i] \cdot B[j] + U$
8: $\quad\quad C[i+j] \leftarrow V$
9: $\quad$ **end for**
10: $\quad C[i+t] \leftarrow U$
11: **end for**
12: RETURN (c)

3.2.4 Reduction

Multiplication and division are the most expensive field arithmetic operations of ECC. In case of a reduction $z \mod p$, where moduli p is not a special form, this reduction can be a very expensive operation as a part of modular multiplication. Expensive operations such as division can be replaced by less-expensive operations with classical reduction methods of Barrett and Montgomery [HMV04, p. 35]. For positive integers p and z, its reduction $z \mod p$ can be solved by using Barret reduction (Algorithm 5) [HMV04].

Algorithm 5 Barret reduction. Adapted from [HMV04]

1: INPUT: $p, b \geq = \lfloor \log_a p \rfloor + 1 \leq z < b^{2k}, and\ \mu = \lfloor b^{2k}/p \rfloor$
2: OUTPUT: $z \mod p$.
3: $\widehat{q} \leftarrow \lfloor \lfloor z/b^{k-1} \rfloor \cdot \mu / b^{k+1} \rfloor$
4: $r \leftarrow (z \mod b^{k+1}) - (\widehat{q} \cdot p \mod b^{k+1})$
5: **if** $r < 0$ **then** $r \leftarrow r + b^{k+1}$
6: **end if**
7: **while** $r \geq p$ **do** $r \leftarrow r - p$
8: **end while**
9: RETURN (r)

3.3 Montgomery Multiplication for Authentication Protocol

Section 2.4.3.9 elaborated the basics of Montgomery multiplication algorithm. This section will provide extra information which was not mentioned and elaborated above in the related work, and is further adapted for an optimized ECC-based authentication protocol in this master's thesis. As mentioned above, the most expensive field arithmetic operations in cryptography are division, multiplication and modulation (reduction), whereas Montgomery is replacing multiplication arithmetic fields with less-expensive inversion op-

erations with a shorter computation time and a very good performance. As a result of a replacement of the huge number of multiplications and divisions, many cryptosystems companies are using Montgomery multiplication algorithm in the solution of their problems. According to [HMV04], Montgomery multiplication algorithm works through transforming inputs. Whereas, this transformation is time critical, and a single modular multiplication cannot afford these expensive transformations. After the transformation of inputs, all of the following multiplications are carried out in the "Montgomery domain". Montgomery multiplication algorithm is also used for exponentiations where some values have to be transformed before many multiplications must be calculated. The advantages of using Montgomery multiplication algorithm are shown in various arithmetic field operations, whereby avoiding divisions is achieved with the help of this algorithm too. For an efficient implementation of a public-key cryptosystem the Montgomery multiplication algorithm is often used.

Montgomery multiplication algorithm can also be used to calculate $c = a \cdot b \mod p$ with the help of the following steps and shown in Algorithm 6 [HMV04, GKPP06]:

- transform a, b to $\bar{a}$, $\bar{b}$
- calculate $\bar{c} = \bar{a} * \bar{b}$; *... Montgomery multiplication
- transform $\bar{c}$ back to c

where p and R must be globally precalculated and $gcd(R, p) = 1$. For the above multiplication, one of the most expensive steps is the reduction *mod p* which is equivalent to a division step.

Algorithm 6 Calculation of $c = a \cdot b \mod p$. Adapted from [HMV04], [GKPP06]

1: $\bar{a} = MontgomeryMult(a, R^2)$
2: $\bar{b} = MontgomeryMult(b, R^2)$
3: $\bar{c} = MontgomeryMult(\bar{a}, \bar{b})$
4: $c = MontgomeryMult(\bar{c}, 1)$
5: RETURN (c)

Algorithm 7 performs a Montgomery multiplication algorithm defined over prime field which is used in this master's thesis for an optimized authentication protocol of a security controller-based application. Special care must be taken on *Line 2* of the Algorithm 7, where more than one arithmetic operation must be calculated. And, due to this complexity as the first operation to be performed is a modulation between p' and R. In order to avoid a division arithmetic operation, which is a time critical and expensive arithmetic operation (*Algorithm 7, Line 2*), another operation for this expensive step is applied. This is achieved and calculated by applying shifting operations, since shifting operations can be performed very quickly in hardware. After the computation of all field arithmetic operations on the *Line 2*, the result r obtained from theses operations has to be compared with the precalculated *p, Line 3.* If the resutl r is bigger than or equal to the precalculated p, then p must be subtracted from *r, Line 4.*

Dhem et al. [DKL$^+$98] performed and elaborated that final subtraction (Algorithm 7, Line 4) depends on the message and the secret bit. They were able to break a 512-bit key in

Algorithm 7 MontgomeryMult(a,b) to calculate $a * b = a \cdot b \cdot R^{-1} \mod p$. Adapted from [HMV04], [GKPP06]

1: $z = a \cdot b$
2: $c = [z + (z \cdot p^{'} \mod R) \cdot p]/R$
3: **if** $c \geq p$ **then**
4: $\quad c = c - p$ **<=Final Subtraction**
5: **end if**
6: RETURN (c)

a few minutes while collecting 300 000 timing measurements. Sato et al. [SST04] described the efficiency of Montgomery multiplication algorithm for public-key cryptosystems, to be dependant on the final subtraction step. Therefore Sato et al. presented a new variant of the final subtraction: $x \cdot y = 3x \mod m$, where m is the underlying modulus.

3.4 Optimized One Way ECC Authentication Protocol

This section describes a security controller authentication protocol based on ECC Diffie-Hellman key exchange. This authentication protocol is a challenge response based on public-key cryptography as shown in Figure 3.7. Furthermore, the software simulation for this authentication protocol is implemented based on elliptic curves defined over prime fields for an optimized ECC-based authentication protocol of a security controller. This master's thesis is based on [BBD$^+$, BHM08], but with a focus of ECC defined over a prime field.

In this master's thesis the public signature key generation, signature verification on an Android NFC-enabled smart phone side, and the signature of security controller is left for future work. The procedure of elliptic curve generation is as follows: let's have an elliptic curve $E = E(a, b)$, with a point $P = (x, y)$ on this curve E, a prime order $q \in N$ and an equation $y^2 = x^3 + ax + b$ defined over elliptic curve E.

The setup initialization of one-way authentication protocol based on Diffie-Hellman key exchange is as follows:

1. **Security controller initialization**
   - Security controller is initialized with a private key ξ_T within interval (0,q)
   - certificate (x_T, s_T)
     - public key x_T which is the affine *x-coordinate* of the point $T = \xi_T \cdot P$
2. **NFC Reader initialization**
   - It is initialized with a public signature key: $PubSKey$. No further keys have to be stored on reader.

The setup interaction of one-way authentication protocol based on Diffie-Hellman key exchange is as follows:

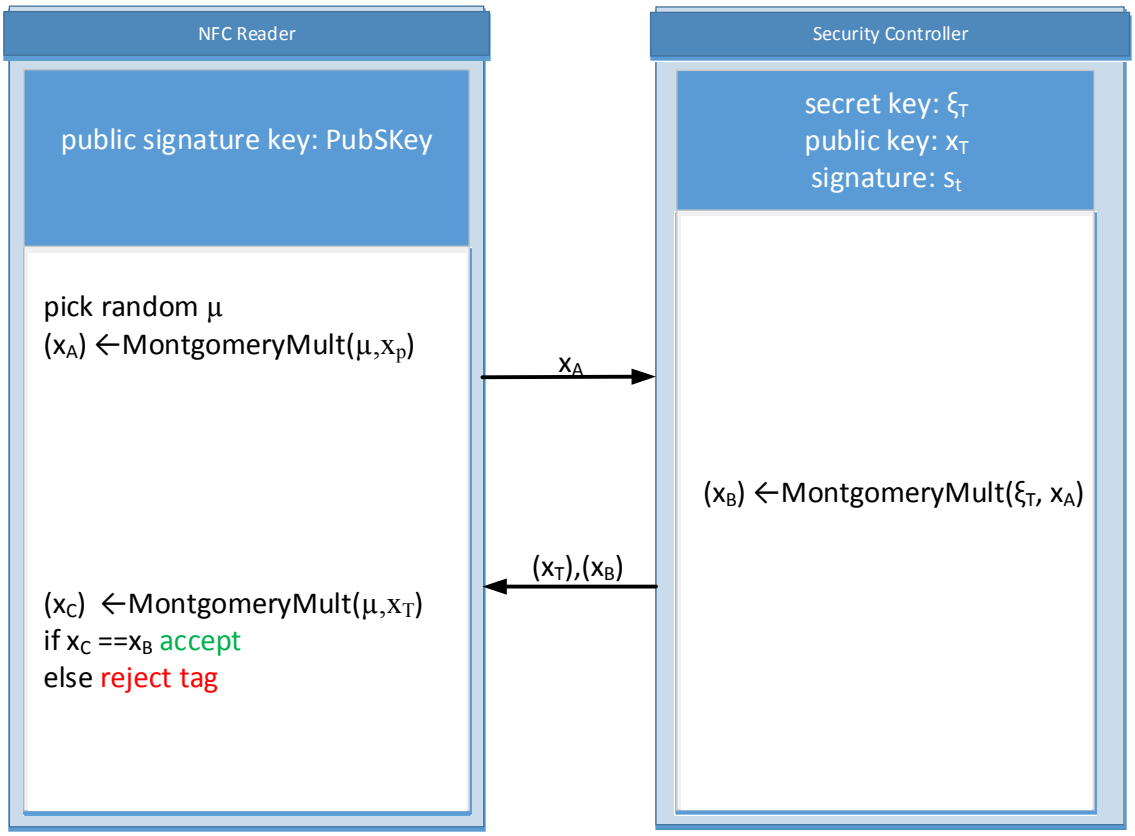


Figure 3.7: Authentication protocol (adapted from [BBD+])

1. **NFC Reader interaction - 1**
   (a) it picks a randomly ephemeral key μ within interval (0,q)
   (b) computes the affine coordinates x_A of the point $A = \mu \cdot P$
   (c) it sends affine coordinate x_A to the security controller
2. **Security controller interaction**
   (a) it calculates $(x_B) \leftarrow MontgomeryMult(\xi_T, x_A)$.
   (b) it sends $x - coordinate\ (x_B), and\ its\ certificate\ (x_T, s_T)$ to NFC reader
3. **NFC Reader interaction - 2**
   (a) $VerifySKey(x_T, s_T)$ verifies tag's certificate (see [BBD+]) ← **is left for future work.**
   - if the signature is invalid then the tag will not be accepted
   - if the signature is valid then the reader verifies response

   (b) by using *Algorithm* 7, it calculates the affine coordinates x_C of the point $C = \mu \cdot (\xi_T \cdot P)$

(c) it checks if $x - coordinates$ (x_C) and (x_B) are equal

- if they are equal, the tag is accepted and declared as an authenticated tag
- if they are not equal, the tag is rejected

Braun et a. [BHM08] showed that integration of an asymmetric cryptographic technique in light-weight devices reduces the cost and provides a higher level of security than the symmetric cryptography. Therefore, Brauen et al. proposed the implementation of asymmetric cryptography for the authentication protocol based on elliptic curves.

The security aspect for an authentication protocol of an RFID tag is nowadays mostly protected by using symmetric cryptosystem. Furthermore, the integration of public-key cryptography is also taken into account. However, the public key cryptography offers applicable advantages for embedded devices although it requires certificate revocation checks. Bocker et al. mentioned that their idea is a continuation of Braun et al. who showed that asymmetric authentication is relevant and suitable to avoid cloning of an RFID tag. Although Falk et al.[FKK+08] presented an application of a passive asymmetric cryptosystem for passive RFID tags, in a high-assurance Avionics multi-domain RFID processing system and different revocation concepts. However, Falk et al. showed that Certificate Revocation List (CRT) technologies are not suitable and good for resource-constraint RFID tags. The implementation of an asymmetric protocol on passive RFID tags is considered to be very complicated due to constraints of an RFID tag e.g. limited power, small size-area, low-power consumption.
Lets suppose an attacker wants to attack this protocol and try to clone an authentic RFID tag. As the first step an attacker tries to achieve, is to get the challenge $A = \mu \cdot P$ from NFC reader. After that this challenge must return a correct response B from RFID tag with an authentic public key $T = \xi_T \cdot P$. If the attacker was able to solve A and T then response B can be solved if and only if an attacker solved the Diffie-Hellman problem as elaborated by Bock et al.

3.4.1 Design of Software Simulation for Authentication Protocol

The first requirement in this master's thesis is the design and implementation of a high-level software simulation application on a PC for the tag one-way ECC-based authentication protocol which is based on elliptic curve Diffie-Hellman key exchange.
The software application at the beginning consists of the simulation system of the Montgomery multiplication algorithm 7 for each of its arithmetic operations. Further, this software simulation is a computer model application that mimics the operation of a real optimized ECC-based authentication protocol between an Android NFC-enabled smart phone and the security controller. As shown in Figure 3.8 *ReferencImplementationP192r1* library consists of two different JAVA entities *MontgomeryMultiplication* and *ECCAuthenticationProtocol*. The aim of this library is to simulate a Montgomery multiplication algorithm for different values and then also the one-way authentication protocol between an reader and security controller for elliptic curves defined over prime field.

In the case of a software simulation, the elliptic curve domain parameters are specified in a *CryptoLibrary* as shown in Figure 3.9. The aim of this library is to construct an elliptic curve with standard domain parameters over F_p (*ECCPointParameters* entity) as described by NIST [fECS00]. *Crypto* JAVA entity generates key pairs for elliptic curves

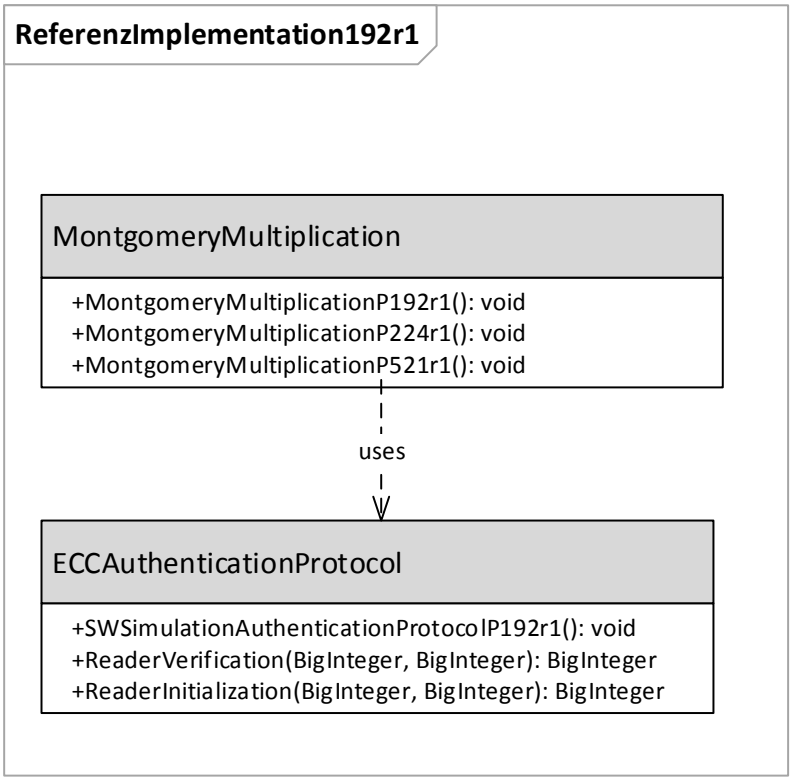


Figure 3.8: Class diagram of the interaction data flow and software simulation

defined over prime field: a private key and a public key which consists of two coordinates (x-,y-coordinates).

The coordination of this software simulation, which mimics communication between an NFC reader and the security controller with the purpose of an authentication protocol, is the same as building an RFID system which performs an authentication protocol built into a software instead of hardware. The security controller does exactly what the reader wants, as simulated in a software application.

A real tag authentication protocol with the help of software simulation is an excellent way of checking if the proposed protocol is suitable and can be used in further steps.
In order to get to the level of security which can achieve high level authentication protocol in RFID security, this software simulation can be tested as a prototype system.
In order to provide a concrete real optimized ECC-based authentication protocol with minimal investment, a software model provides the option to change the precalculated data (see section 3.4) with a wide range of possibilities.

Furthermore, the software simulation provides the conceptual solution which is necessary to design and implement the following two steps: **Multiple APDUs Interaction** 3.4.2 and **Single APDU Interaction** 3.4.3. In order to predict the behavior of the software simulation, the steps mentioned above are preliminaries for a correct and optimized authentication protocol.

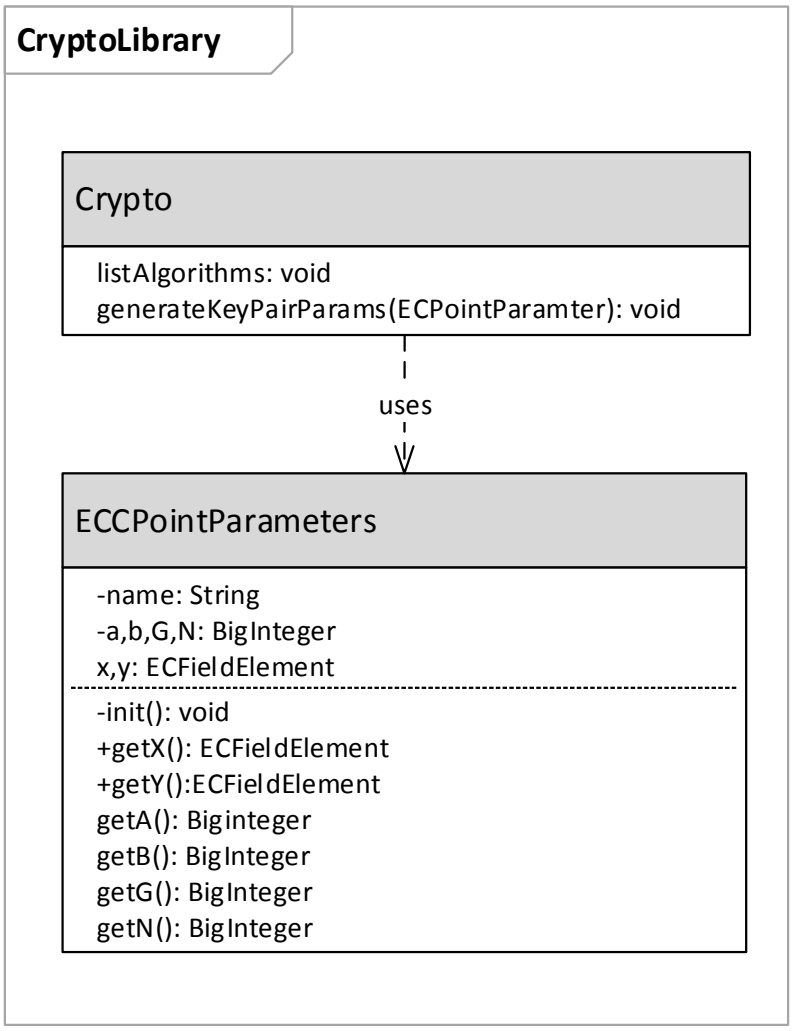


Figure 3.9: Crypto Library: Class Diagram

3.4.2 Design of Authentication Protocol with Multiple APDUs

As elaborated in Section 3.4.1, a software simulation provides an efficient method and a very helpful prototyping solution for an optimized authentication protocol. This authentication protocol is applicable by interaction between an Android NFC-enabled smart phone and the security controller over an NFC interface, whereby the messages are exchanged by using APDU. Section 3.3 describes that an applicable and optional algorithm for an optimized authentication protocol is a Montgomery multiplication algorithm defined over ECC prime field.

An authentication protocol within this section is realized consisting of the following steps: firstly a reader initialization occurs where the challenge x_A is computed, then a Montgomery multiplication algorithm on the security controller is computed by calculating separately each of its field arithmetic operations with the help of multiple APDUs and finally a reader verification follows. For the parameters which are not in the Montgomery

domain their transformation is taken into account.

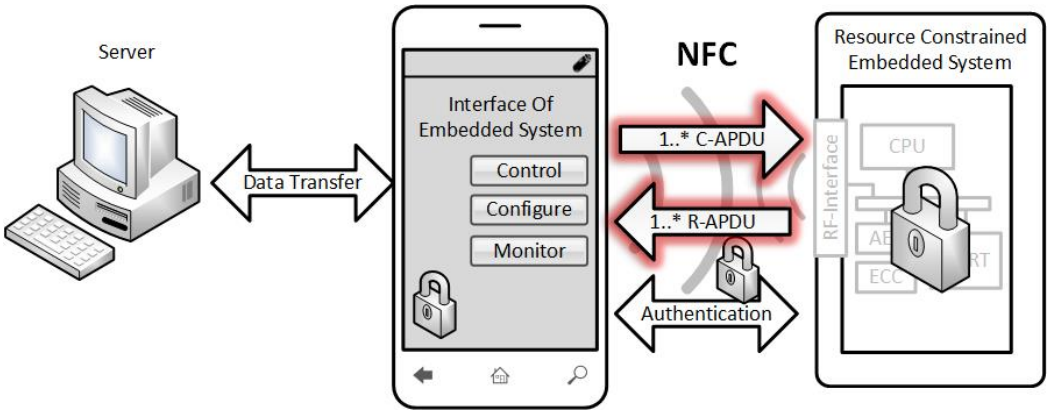


Figure 3.10: Multiple APDUs interaction system overview: authentication protocol by using multiple APDUs

Authentication of an RFID tag can potentially be performed in different ways. In this master's thesis a one-way authentication protocol is enforced from an NFC reader instead of enforcing from a software application on a PC. This authentication protocol has to be hardwired both into the RFID system (NFC reader) and security controller, whereby the security controller must always process data for an NFC reader for the purpose of an optimized authentication protocol before transmitting.

For an optimized authentication protocol, a security controller receives different requests from an Android NFC-enabled smart phone to authenticate itself to a reader.
As shown in Figure 3.10 an optimized ECC-based authentication protocol for security controller-based applications is achieved with the help of multiple APDUs, whereby more than one C-APDUs is sent from an Android NFC-enabled smart phone to the security controller over an NFC interface. And for each of the C-APDUs, a response R-APDU is computed and sent back from the security controller to an Android NFC-enabled smart phone whereby each of the R-APDUs consist of various information depending on the request C-APDU sent from the NFC reader.
C-APDU consist of the following field arithmetic operations:

- *Multiplication*,
- *Division*,
- *Addition*,
- *Substraction*,
- *Reduction*,
- ...

Infineon security controller provides all of the above mentioned field arithmetic operations defined over prime field, and many other operations. There are special APDUs

implemented on the security controller within this project for these field arithmetic operations, which are necessary for an optimized one-way authentication protocol.

It should be ensured at the design and manufacturing phase that the security controller provides these arithmetic field operations.
In order to check the correctness of a security controller, each of the R-APDUs which are a single arithmetic operation of the Montgomery multiplication algorithm must be compared with the results obtained from software simulation. In this way an authentication protocol of a security controller to an Android NFC-enabled smart phone is achieved with the help of multiple APDUs Interaction.
Figure 3.11 shows the essential steps corresponding to an authentication protocol between

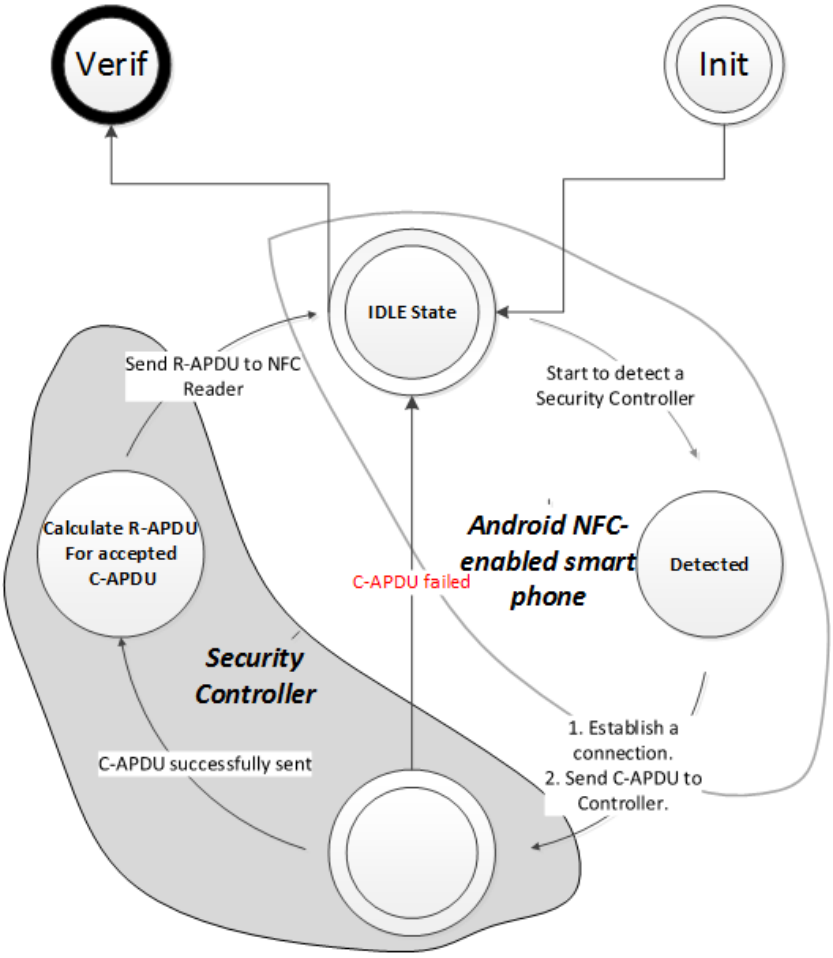


Figure 3.11: State Diagram: NFC-reader to Security Controller

an Android NFC-enabled smart phone side and the security controller's side: starting from the initialization, detection of a security controller, a connection establishment between a reader and security controller, APDU preparation, sending C-APDU to the security controller and towards the response R-APDU from the security controller to an Android NFC-enabled smart phone for a verification. It is very important to note that an Android NFC-enabled smart phone has to be an authorised reader to the security controller, before

any important data exchange for any purpose between them occurs.

After an authorization of an NFC reader, the data exchange between the security controller and an Android NFC-enabled smart phone is permitted. The authentication protocol mechanism simulated by software application is an excellent reference for an optimized ECC-based authentication protocol between an Android NFC-enabled smart phone and the security controller. Although before transmission of any data to the security controller occurs, a processing of the data is required. Therefore, it is necessary to implement a Montgomery multiplication algorithm 7 on the NFC-reader and the result computed from Montgomery multiplication is further sent as a challenge to the security controller.

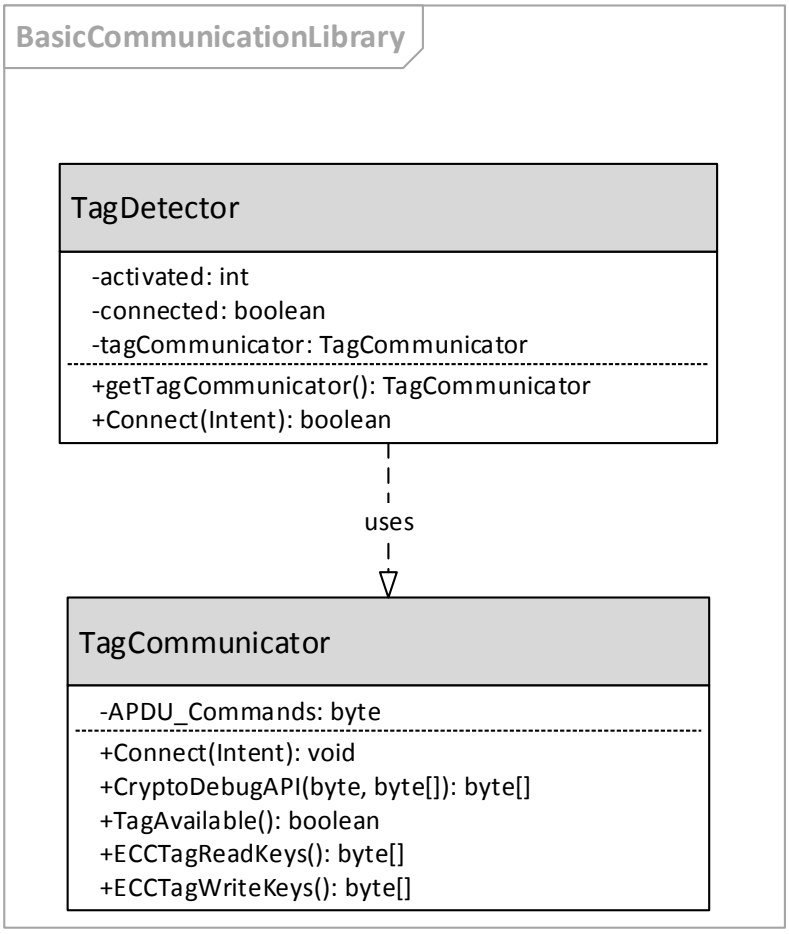


Figure 3.12: Basic Communication Library: Class Diagram

As mentioned above the communication establishment between an Android NFC-enabled smart phone and the security controller is achieved by using *BasicCommunicationLibrary* which provides two different entities as shown in Figure 3.12. *TagDetector* JAVA entity's aim as the main instance of the *BasicCommunicationLibrary*, is to detect a security controller. *TagCommunicator* JAVA entity's duty is to establish a connection to a security controller for the purpose of sending and receiving APDUs.

Figure 3.13 shows the class diagram of a *Multiple APDUs Interaction* option of an optimized ECC-based authentication protocol. As elaborated in the Section 3.1, *MainUI*

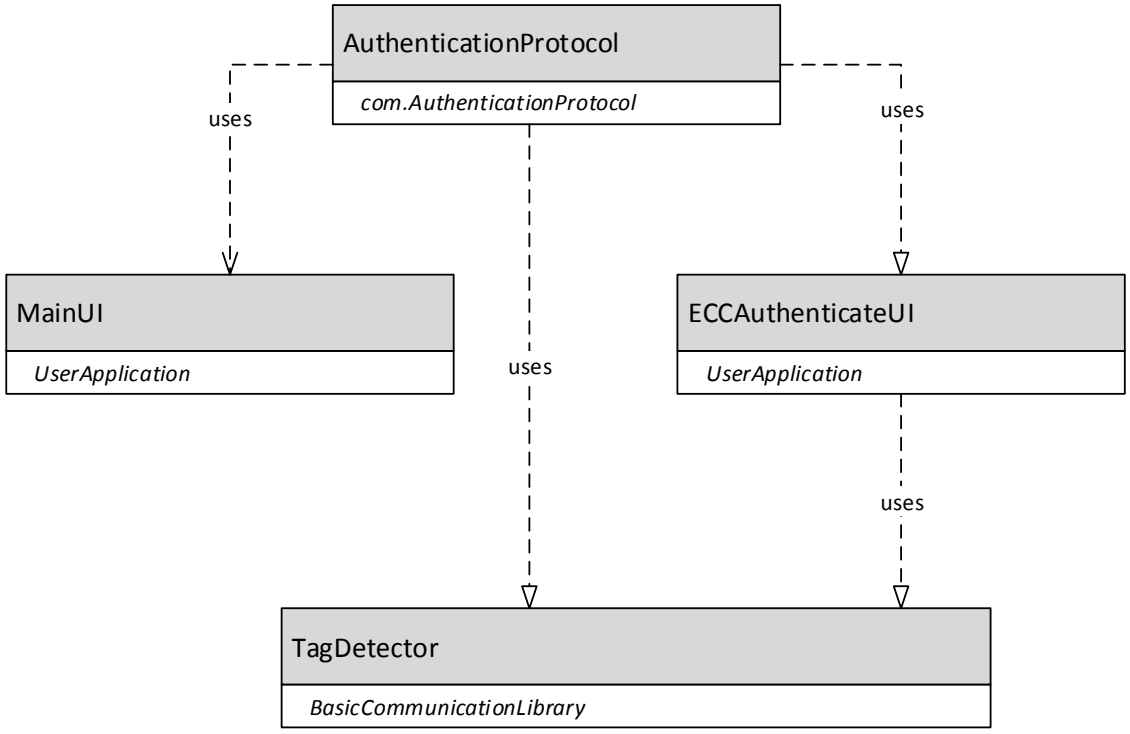


Figure 3.13: Multiple APDUs Interaction: Class Diagram

entity is the main class of *UserApplication* library, where the initialization of the application for the purpose of an authentication protocol occurs.

ECCAuthenticateUI entity with the help of widgets provided by Android OS helps to prepare a request for the security controller e.g. field arithmetic operations, Montgomery multiplication algorithm. This entity references to an instance of *TagCommunicator* which initiates an interaction with the security controller. Section 3.3 described that for the purpose of an optimized ECC-based authentication protocol, the computed affine x-coordiante x_A by Montgomery multiplication algorithm is sent from the NFC-reader to the security controller.

After computation of a Montgomery multiplication algorithm on the security controller with the help of multiple APDUs, the final result that is obtained from the controller is further used on an NFC-reader for the verification of authenticity as shown in Figure 3.7.

3.4.3 Design Authentication Protocol with Single APDU

This Section is a continuation work of the section above, whereby a Montgomery multiplication algorithm on the security controller is computed with the help of a single APDU, sent from an Android NFC-enabled smart phone.
The aim of an optimized ECC-based authentication protocol is to ensure the data exchange between an NFC-enabled smart phone and a security controller by integrating an authentication protocol. As shown in Figure 3.7 a single point multiplication is computed

on the security controller. Due to the single point multiplication on the security controller, a high performance can be achieved for an optimized ECC-based authentication protocol. Furthermore, as shown in Figure 3.14 an authentication protocol is applied by

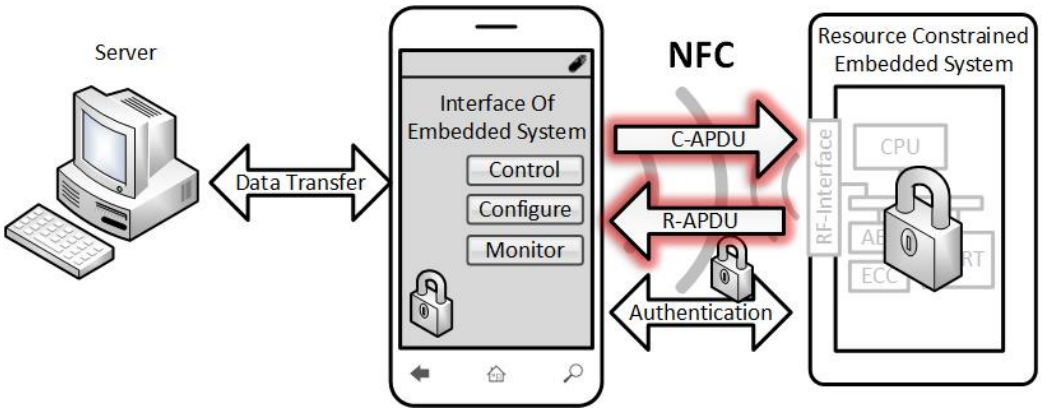


Figure 3.14: Single APDU interaction system overview: authentication protocol by using a single APDU

sending a single C-APDU from an Android NFC-enabled device to the security controller over an NFC interface and then receiving a single R-APDU from the security controller. In Section 3.4.2 for the purpose of an optimized ECC-based authentication protocol, a Montgomery multiplication algorithm on the security controller is calculated by sending multiple APDUs from an Android NFC-enabled smart phone.

For each of its field arithmetic operations, a message is saved in the form of bytes block in a single C-APDU and sent from the Android NFC reader to the security controller. Security controller now provides a single command, which is dedicated to the authentication protocol. Within this command, a Montgomery multiplication is computed with its Montgomery domain parameters. As mentioned in Section 3.4 there are some necessary precalculated parameters to compute Montgomery multiplication algorithm and these parameters must be stored on the security controller's side. The Android NFC-enabled smart phone computes the affine x-coordinate x_A (reader initialization) and sends it as a challenge to the security controller. The security controller receives this challenge as a request from the NFC reader saved in a special APDU command. It calculates the response and saves it in an R-APDU.

On the other side the NFC-reader receives a single R-APDU from the security controller. This R-APDU is further used for the validation of an authenticity on the Android NFC-enabled smart phone (reader verification).

Chapter 4

Implementation

This chapter gives an overview of the implementation phase of an optimized ECC-based authentication protocol for security controller-based applications, where various steps are described and followed. Figure 4.1 shows the steps followed in the design phase which are essential for application in the implementation phase.

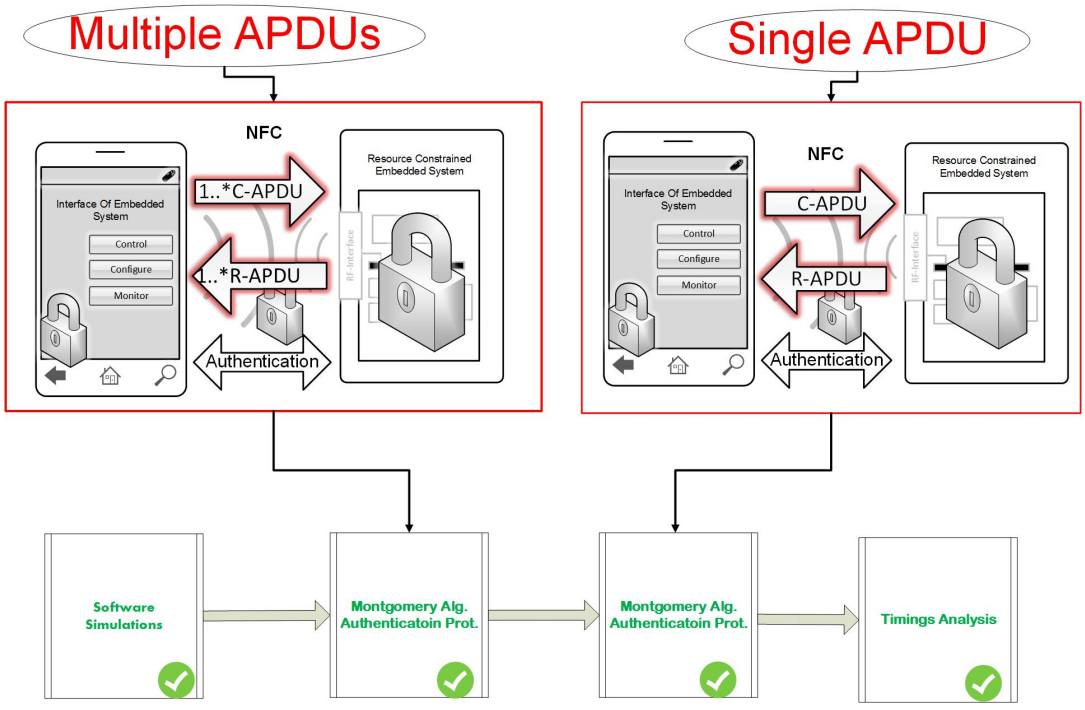


Figure 4.1: Step by Step Design and Implementation Phase

At the beginning of this master's thesis literature had to be reviewed, with the purpose of collecting various ideas. Based on various ideas and solutions, a unique protocol with its specifications was implemented. After a related work research phase, a software was designed and implemented, which simulates an optimized authentication protocol based on Diffie-Hellman key exchange as shown in Figure 4.1. Furthermore, the results obtained

from the software simulation were applied on a real authentication protocol between an NFC-reader and the security controller. However, before applying a real authentication protocol, a connection must be established between an Android NFC-enabled smart phone and the security controller over NFC-interface for a data exchange. An optimized ECC-based authentication protocol was implemented following the steps in Figure 4.1, whereby firstly, more than one APDU is sent from the reader to the security controller over an NFC interface, separately for a data exchange. After that, a protocol was achieved with the help of a single APDU interaction between reader and the security controller.

Finally, a timing analysis for this authentication protocol was carried out, and the results obtained in this project were compared to approaches in related work.

4.1 Development Environment

Software simulation of an authentication protocol is implemented using Netbeans IDE 7.4 [Orab]. An initiator in the implementation phase of this project is chosen as a smart phone Samsung Galaxy i9300 SIII[1], NFC-enabled device running Android OS version 4.3, running CPU: Quad-core 1.4 GHz Cortex-A9 with a Chipset: Exynos 4412 Quad.

Samsung SIII NFC-enabled smart phone is chosen due to its open source operating system and a support of ECC cryptographic libraries, which is suitable for our project.

Authentication protocol for Android NFC-enabled smart phone is implemented by using an Eclipse IDE, Android SDK 4.3 and JDK7. Developing any pure software for Android is possible by using JAVA Object Oriented Programming language (OOP).

The results obtained by the software simulation and NFC reader to security controller interaction e.g. timing analysis of field arithmetic operations, Montgomery multiplication algorithm, authentication protocol with various variants are ploted Matlab R2010b, developed by MathWorks [Mat].

4.2 Smart Phone and Security Controller Interaction

This section shows the essential conditions required to establish a connection between an Android NFC-enabled smart phone and the security controller, which are further necessary for an implementation of an optimized ECC-based authentication protocol. As elaborated in Section 3.1 the *BasicCommunicationLibrary* consists of all the necessary entity classes for a connection establishment over an NFC interface between an Android NFC-enabled smart phone and the security controller and the methods provided by those entities are taken with permission from ITI[2] (Norbert Druml) which were also used by Bashagic [Bas13] in his master's thesis.

[1] http://www.gsmarena.com/samsung_i9300_galaxy_s_iii-4238.php
[2] Institute for Technical Informatics

The following basic NFC entities provided by Android OS are used within this project to send and retrieve NFC data:

- `android.nfc.NfcAdapter`
- `android.nfc.Tag`
- `android.nfc.tech.IsoDep`

TagDetector is used for the detection of the security controller; however this entity is called whenever an Android NFC-enabled smart phone attempts to detect the security controller for a purpose of an authentication protocol.
TagCommunicator entity with the detected security controller establishes a connection, communication, and enables the data exchange between the Android NFC-enabled smart phone and security controller with the help of `CryptoDebugAPI(byte, byte[])` method.

This function within this entity is used for sending C-APDU to and receiving R-APDU from the security controller. However, due to its access constraints a user application cannot directly access the low level *TagCommunicator* entity class. Therefore, *TagDetector* is the main entity of *BasicCommunicationLibrary*, whose instance can be generated by user application and with the help of this generated instance, the access to low-level entity *TagCommunicator* is allowed.
The low-level entity class *TagCommunicator*, which initiates the interaction between an Android NFC-enabled smart phone and the security controller over NFC interface, provides furthermore, some specific methods which are necessary for a connection establishment. The preliminary condition of the *BasicCommunicationLibrary* is after the detection of the security controller, to achieve a communication establishment between the NFC reader and the security controller. Whenever, the necessity of the data exchange appears, an attendance of the detected (connected) security controller is required by using `TagAvailable()` method as shown in the Listing 4.1. Before reading and writing to an NFC tag occurs, the opening of communication with the tag is necessary. This is possible by using the method `connect()` provided by Android OS. APDU command is sent to the security controller and a response (R-APDU) is received with the help of `transceive( byte[] data)` method provided by Android OS. Appendix of SoD (length) or EoD (CRC) to the payload in application is not necessary, it will be automatically calculated. If the security controller is not within the filed, an exception is thrown. As soon as the security controller is detected the next steps of connection/communication establishment and reading of *General Information Record* are available e.g. the version of the current solution on the transponder's side. It is very important to mention that if the security controller is not compatible with the Android NFC-enabled smart phone, or available, the steps mentioned above are not further performed or executed.

Listing 4.1: TagDetector: Detection of Security Controller

```
1  /*
2   * Test if the tag is available (polling from outside)
3   */
4  public boolean TagAvailable() {
5
6    IsoDep tag = IsoDep.get(this.connectedTag);
```

```
7
8     try {
9       tag.connect();
10
11      boolean connected = tag.isConnected();
12
13      tag.transceive(new byte[] { });
14
15      tag.close();
16
17      return connected;
18    } catch (Exception e) {
19      Log.v("TagCommunicator", "TagAvailable failed (Card unavailable)");
20    }
21
22    return false;
23  }
```

The two following sections 4.4 and 4.5 which are applied for an optimized ECC-based authentication protocol do not require the retrieval of the NDEF information stored on the security controller. Therefore, retrieving the *General Information Record* is not included within the request of connection.

Listing 4.2: TagCommunicator: Connection Extablishment

```
1   /*
2    * Connect to the tag
3    */
4   public void Connect(Intent intent) {
5
6
7     // display the invoked command
8     Log.v("TagCommunicator", "Connect" + intent.toString() + ")");
9
10    try {
11
12      // get the detected security controller
13      Tag tagFromIntent = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
14
15      this.connectedTag = tagFromIntent;
16
17    } catch (Exception e) {
18      Log.e("TagCommunicator",
19          "Connect failed (Card unavailable or incompatible)");
20    }
21  }
```

An instance of the *TagCommunicator* low-level entity class as mentioned above is initialized within *TagDetector* entity class. This provides an access to NFC chip, and sending C-APDU over NFC interface from an Android NFC-enabled smart phone to the security controller is realized. After a successful detection of the security controller, the connection and communication establishment with the controller is possible by using `Connect()`

method provided by the entity *TagCommunicator* as shown in the Listing 4.2. Those are the necessary methods used in this master thesis, for the interaction between an Android NFC-enabled smart phone and the security controller.

4.3 Implementation of Software Simulation

As elaborated in Section 2.4 an authentication protocol could be realised in the following ways: symmetric and asymmetric protocols. Using a symmetric protocol for an authentication of the tag has some major draw-backs. In this case the tag's secret key must be stored inside an Android NFC-enabled reader. Furthermore, a secure communication between NFC-reader and back-end database storing these keys must be provided. However, during the implementation of this project there is no secure communication provided between an Android NFC-enabled smart phone and back-end database. Due to this reason the idea within this master thesis is to apply an asymmetric protocol for the authenticity of the tag. In the case of an asymmetric protocol no secret keys must be stored inside a reader, and no secure communication is necessary between an Android NFC-enabled smart phone and back-end database. Therefore, an asymmetric protocol provides a random generation of keys, which makes it difficult for an attacker to threaten and obtain these keys, and clone the RFID tag. Integration of an asymmetric protocol into such low devices is a technological challenge due to the resource-constraints of an RFID tag.
First, in this master's thesis for the purpose of an authentication protocol a software simulation of a Montgomery multiplication algorithm 7 is implemented and simulated. Its field arithmetic field operations are tested with various length of operands. After a successful simulation of a Montgomery multiplication algorithm for different elliptic curves defined over prime field (*secp192r1, secp224r1*, *secp521r1*), a software simulation is implemented and simulated for an optimized ECC-based authentication protocol. The mandatory construction of *secp192r1* elliptic curve is shown in Figure 4.2. Various test cases within software simulations are taken into account of an authentication protocol. The essential transformation of inputs for Montgomery multiplication is achieved with the help of Montgomery multiplication algorithm 7. However, as described at the beginning of this chapter the practical part of this master thesis is mainly realized by using JAVA programming language. This OOP language is provided with various cryptography libraries which emphasize security, including language safety, cryptography, public key infrastructure, authentication protocol, secure communication, and access control [Doc].

Listing 4.3: Bouncy Castle support of cryptographic algorithms for Android. Adapted from. [Ele13]

```
1 BC/BouncyCastle Security Provider v1.46/1.460000
2   KeyAgreement/ECDH
3   KeyFactory/EC
4   KeyPairGenerator/EC
5   Signature/ECDSA
6   Signature/NONEwithECDSA
7   Signature/SHA256WITHECDSA
8   Signature/SHA384WITHECDSA
9   Signature/SHA512WITHECDSA
```

Bouncy Castle is one out of many open source JAVA libraries which supports EC and is used for cryptographic functions. Android used this open source library up to a certain version, but due to its minor support of ECC algorithms as shown on Listing 4.3, switched to the usage of another open source Java library for cryptography called *Spongy Castle* [Tyl].

The reason for supporting so few algorithms by the *Bouncy Castle* library, is due to its class name's conflict with Android's operating system. Therefore, *Spongy Castle* is the newer version of *Bouncy Castle* which supports more algorithms as shown in the Listing 4.4, and works in Android OS for a cryptographic functionality with some small changes.

Listing 4.4: Spongy Castle support of cryptographic algorithms for Android. Adapted from. [Ele13]

```
SC/BouncyCastle Security Provider v1.46/1.460000
   AlgorithmParameters/SHA1WITHECDSA
   ...
   KeyAgreement/ECDH
   KeyAgreement/ECDHC
   KeyAgreement/ECMQV
   KeyFactory/EC
   KeyFactory/ECDH
   KeyFactory/ECDHC
   KeyFactory/ECDSA
   KeyPairGenerator/EC
   KeyPairGenerator/ECDH
   KeyPairGenerator/ECDHC
   KeyPairGenerator/ECDSA
  ...
```

However, nearly everything is left the same as in the *Bouncy Castle* library, only the following changes are made:

- in order to avoid the conflicts all the packages are renamed from **org.bouncycastle.*** to **org.spongycycastle.***
- the Java Security API Provider name is now changed from **BC** to **SC**
- `BouncyClassProvider` class remained the same, although moved to the package **org.spongycastle.provider**

In order to use all of the algorithms in the Listing 4.4 and some others not listed above, it is necessary to define this provider in the application as follows [Tyl].

```
static {
    Security.insertProviderAt(new org.spongycastle.jce.provider.
        BouncyCastleProvider(), 1);
    }
```

Although a very difficult decision appeared on integrating and using elliptic curves defined over binary fields or prime fields for embedded devices. Through reviewing related work's ideas and solutions, we came across the fact that elliptic curve defined over prime field is

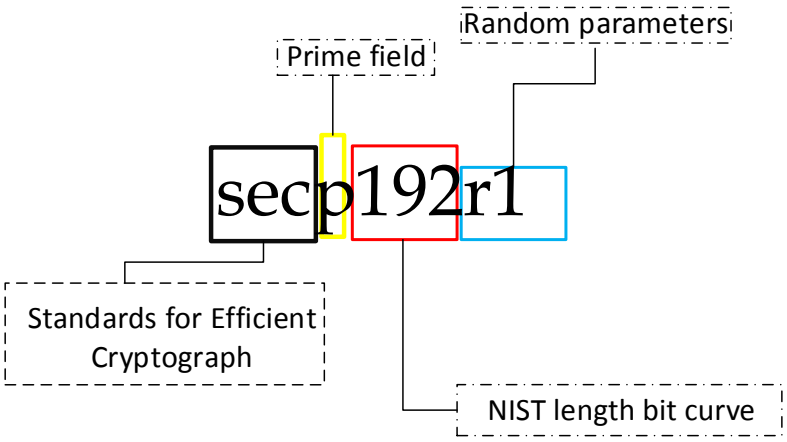


Figure 4.2: NIST: Elliptic curve 192-bit length (adapted from [fECS00])

more suitable than elliptic curve defined over binary field, for an ECC-based authentication protocol for security controller.

The other reason for using curves defined over prime fields in this master thesis is the support of SLE hardware multiplier by the security controller. Spongy Castle open source library on Android OS supports the following curves defined over prime field: *secp112r1*, *secp128r1*, *secp160r1*, *secp192r1*, *secp224r1*, *secp256r1*, *secp384r1*, and *secp521r1*.
For an optimized ECC-based authentication protocol for security controller-based application, the state of the art best curve with a very good level of security is selected *secp192r1*. Due to the support of SLE by security controller the 192-bit curve is selected. Although, the authentication protocol is also simulated for other curves defined over prime field and a binary field.

Representing elliptic curve domain parameters is provided through various ways, for example using ASN.1 syntax. Where for use in X.509 certificates and elsewhere the following syntax is recommended [Ele13]:

```
ECGenParameterSpec   ecGenSpec = new ECGenParameterSpec("secp192r1");
```

Elliptic curve domain parameters for *secp192r1* can be described as follows and are implemented within *CryptoLibrary* as shown in Figure 3.9 [Ele13]:

```
EllipticCurve curve = new EllipticCurve(
      new ECFieldFp(
new BigInteger("FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF")), //q
new BigInteger("FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFC", 16), // a
new BigInteger("64210519E59C80E70FA7E9AB72243049FEB8DEECC146B9B1", 16)); // b

ECParameterSpec ecSpec = new ECParameterSpec(
      curve,
      ECPointUtil.decodePoint(curve, Hex.decode("03188
          DA80EB03090F67CBF20EB43A18800F4FF0AFD82FF1012")), // G
      new BigInteger("FFFFFFFFFFFFFFFFFFFFFFFF99DEF836146BC9B1B4D22831"), // n
           1); // h
```

ECC key pairs generation is possible only by providing the algorithm and the provider as follows [Ele13]:

```
KeyPairGenerator g = KeyPairGenerator.getInstance("ECDH", "SC");
```

PrivateKey and *PublicKey* are also generated from the above specified *KeyPairGenerator*, and *EllipticCurve*. Affine x-coordinate of *PublicKey* is a point on the elliptic curve defined over prime field. Elliptic curve domain parameters are defined by using Java class **BigInteger** [Oraa]. BigInteger class provides all the necessary arithmetic operations analogues to Java's primitive integers operators. It provides one of the most expensive operators in cryptography: **modular arithmetic operations**. Furthermore, shifting operators, exponentiation, comparison operators etc. are provided by this JAVA class.
BigInteger class provides various constructors, although for ECC is as follows:

```
BigInteger(String val, int radix)
```

which is used to translate the String representation of a BigInteger in the specified radix into a BigInteger. Radix of the String representation consists of the following values:

- 2 - binary value
- 8 - octal value
- 16 - hexadecimal value

whereas in this project a hexadecimal value is applied for the translation of the string representation into a BigInteger.

The field arithmetic operations used for elliptic curve operations defined over binary and prime field, differ from each other in the implementation form, which has a huge impact on the processing time of these operations [GKPP06]. Micro-processor architecture is the main dependency of the field arithmetic algorithm's efficiency. However, in this section the software implementation of these efficient algorithms such as addition, subtraction, multiplication, and inversion that are used for the elliptic curve operations defined over prime fields are discussed, and these algorithms are chosen carefully, considering the constraints on the available resources. The most used field arithmetic operations in the finite field of cryptography are operations with odd primes. Furthermore, the hardware implementation of the field arithmetic operations is available in the book [HMV04, Chapter. 5].

4.3.1 Addition

Modular addition algorithm 8 is given in terms of *Multi-precision addition* (Algorithm 2) which performs an adaption of the algorithm 2 with an additional step for reduction modulo p [HMV04, Chapter. 2].

Modular addition algorithm works bitwise. There are some programming languages where **BigInteger** class is not provided. Due to this lack, BigInteger must be manually implemented, and in case of accurateness the modular addition Algorithm 8 must be implemented. However, in this master thesis, the software implementation of these efficient algorithms is realized by using JAVA OOP language, where the **BigInteger** class is provided. For example, addition of two Bigintegers is as follows:

Algorithm 8 Addition in F_p. Adapted from [HMV04]

1: INPUT: *Modulus* p, *and the integers* $a,\ b \in [0, p-1]$
2: OUTPUT: $c = (a + b) \mod p$
3: *Use Algorithm* 2 *to obtain* (ϵ, c)
4: **if** $\epsilon = 1$ **then** *substract* p *from* $c = (C[t-1], ..., C[C2], C[1], C[0])$;
5: **else if** $c \geq p$ **then** **c** $\leftarrow$ **c-p**
6: **end if**
7: RETURN (c)

```
BigInteger a = new BigInteger("012345678ABCDEF012345678ABCDEF", 16);
BigInteger b = new BigInteger("012345678ABCDEF012345678ABCDEF", 16);
BigInteger result = a.add(b); // a+b
```

The addition modular arithmetic operations provided by JAVA **BigInteger** [Oraa] fulfills the computation of the modular addition Algorithm 8 for prime field.

4.3.2 Subtraction

The same procedure also follows for modular subtraction in F_p (Algorithm 9), where multi-precision subtraction (Algorithm 3) is further adapted. After that, it is followed with an subtraction of p, if the result obtained from Algorithm 3 is bigger or equal to prime p.

Algorithm 9 Substraction in F_p. Adapted from [HMV04]

1: INPUT: *Modulus* p, *and the integers* $a,\ b \in [0, p-1]$
2: OUTPUT: $c = (a - b) \mod p$
3: *Use Algorithm* 3 *to obtain* (ϵ, c)
4: **if** $\epsilon = 1$ **then** *substract* p *from* $c = (C[t-1], ..., C[C2], C[1], C[0])$;
5: **else if** $c \geq p$ **then** **c** $\leftarrow$ **c-p**
6: **end if**
7: RETURN (c)

The modular subtraction in F_p (Algorithm 9) in this master thesis for an optimized ECC-based authentication protocol is also achieved by using a modular arithmetic operation provided by JAVA **BigInteger** class as follows:

```
BigInteger a = new BigInteger("012345678ABCDEF012345678ABCDEF", 16);
BigInteger b = new BigInteger("012345678ABCDEF012345678ABCDEF", 16);
BigInteger result = a.subtract(b); // a-b
```

4.3.3 Multiplication

One of the most expensive modular arithmetic operation in ECC is a modular multiplication operation. However, multi-precision integer multiplication or squaring are used to perform a modular multiplication, squaring and then reducing the double-bit length [GKPP06]. Although, modular multiplication or squaring are also achieved in this project

by using modular arithmetic multiplication operation provided by JAVA **BigInteger** class, as follows:

```
BigInteger a = new BigInteger("012345678ABCDEF012345678ABCDEF", 16);
BigInteger b = new BigInteger("012345678ABCDEF012345678ABCDEF", 16);
BigInteger result = a.mulitply(b); // a * b
```

4.3.4 Other Modular Arithmetic operations

The other efficient modular arithmetic operation algorithms defined over prime field are also achieved by using operations for modular arithmetic provided by **BigInteger**. The modular division arithmetic operation is achieved by using the following shifting modular arithmetic operation:

```
BigInteger a = new BigInteger("012345678ABCDEF012345678ABCDEF", 16);
BigInteger b = new BigInteger(2);
BigInteger result = a.shifRight(b); // a >> b
```

The arithmetic operation `mod` also known as the final reduction in Montgomery multiplication (Algorithm 7 *Line 2*) is also provided, and it is implemented as follows:

```
BigInteger a = new BigInteger("012345678ABCDEF012345678ABCDEF", 16);
BigInteger b = new BigInteger(2);
BigInteger result = a.mod(b); // a mod b
```

This modular arithmetic operation provided by the **BigInteger** class, differs from *remainder* and always returns a *non-negative* BigInteger.
Further information on modular arithmetic operations provided by JAVA **BigInteger** is available at [Oraa].

4.3.5 Simulation of Montgomery Multiplication Algorithm over Prime Field F_p

`MontgomeryMultiplicationP192r1(BigInteger, BigInteger)` method Listing 4.5 provided by the *MontgomeryMultiplication* entity as shown in Figure 3.8, performs a simulation of a Montgomery multiplication algorithm for elliptic curves defined over prime field e.g. *secp192r1*.

Listing 4.5: Software Simulation: Montgomery Multiplication Algorithm 7

```
public static BigInteger MontgomeryMultiplicationP192r1(BigInteger a, BigInteger
    b)
  {
    BigInteger res = null;

    // z = a*b
    BigInteger z = a.multiply(b);


    // ------------------------------------------------
    // -------z = ( z + ((z*p') mod R)*p ) / R------------
    // ------------------------------------------------
```

```
12
13      // zp_tiled = z*p'
14      BigInteger zp_tiled = z.multiply(p_tiled);
15
16      // v = zp_tiled mod R
17      BigInteger v = zp_tiled.mod(R);
18
19      // vp = v*p
20      BigInteger vp = v.multiply(p);
21
22      // w = vp + z
23      BigInteger w = z.add(vp);
24
25      // u = w/R
26      BigInteger u = w.shiftRight(BITS_PER_WORD * WORDS_PER_BIGINT);
27      BigInteger mod = null;
28      mod = p;
29
30      // ----------------------------------------------------
31      // ------ c = c >= p ? c-p : c ------------------------
32      // ----------------------------------------------------
33      int comp = u.compareTo(mod);
34
35      if (comp >= 0) {
36        c = c.subtract(mod);
37      } else {
38        BigInteger dummy = c.subtract(mod);
39      }
40
41      res = c;
42      return res;
43    }
```

The `dummy` variable in the *Line 34* is more or less only to balance the timing analysis but it does not impact the authentication protocol. The condition `if` statement *Line 31*, is one of the weakest points for Montgomery multiplication algorithm, where the attackers can eavesdrope the private key. The subtraction within this condition is executed depending on the secret bit and the message

4.3.6 Simulation of Authentication Protocol over Prime Field

The simulation of a real authentication protocol for *secp192r1* elliptic curve between an Android NFC-enabled smart phone and the security controller is performed by the invocation of the method `SWSimulationAuthenticationProtocolP192r1` provided by the entity *ECCAuthenticationProtocol*, whereby the transformation of the operands is taken into account of software simulation as shown in Figure 3.8. Therefore more than three operations are performed due to the transformation necessity.

This means that for every multiplication on the Android NFC-enabled smart phone and the security controller for the aim of an optimized ECC-based authentication protocol the steps in Algorithm 6 are followed.

Spongy Castle Library is used to generate an elliptic curve with its recommended parameters, furthermore, to also choose a point x_P on the elliptic curve in an affine representation. The randomly ephemeral key on the NFC reader is also generated by using Spongy Castle **(SC)** library. **SC** is further used to generate the security controller's private key and out of this key to generate its public key which is an affine coordinate.

4.3.7 Simulation of Authentication Protocol over Binary Field

The source code for the software simulation of one-way authentication protocol for elliptic curves defined over binary field is based on the master thesis written by Höller [Höl13], although it is converted to JAVA programming language in order to simulate it on an Android NFC-enabled smart phone.

Class Diagram that are used for a one-way authentication protocol of elliptic curves defined over binary field is shown in Figure 4.3. *Main* JAVA entity is the main component

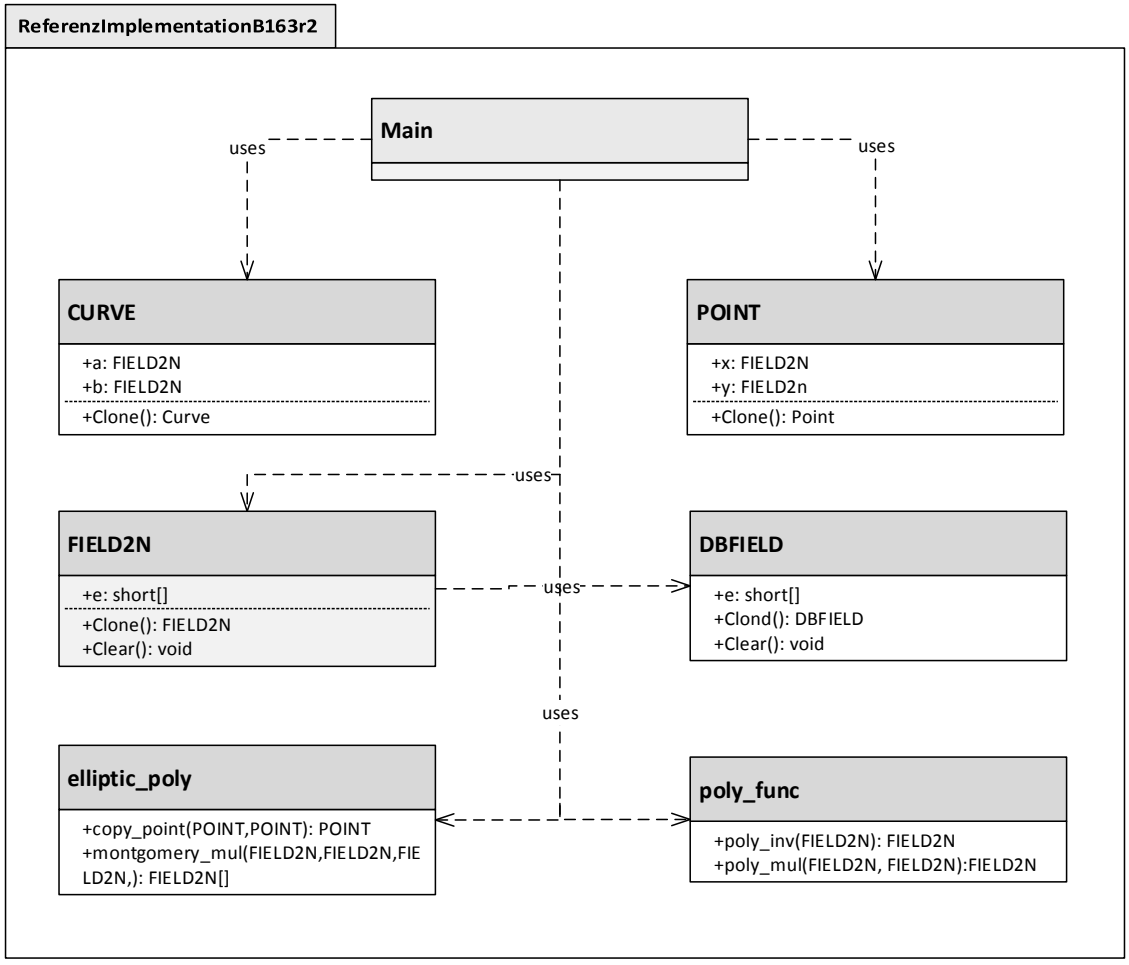


Figure 4.3: Reference Implementation for Binary Field (adapted from [Höl13])

of an authentication protocol defined over binary field, which holds the essentials such as

point elliptic multiplication, Montgomery multiplication algorithm 1, reader initialization, and verification of an authenticity. *CURVE* and *POINT* entities are used to create elliptic curve points and parameters. Further information about the other entities is available in the book *Implementing Elliptic Curve* [Ros99] and also an open source code is available online at [Ros].

4.4 Implementation of Authentication Protocol with Multiple APDUs

This section performs an optimized ECC-based authentication protocol between a reader and the security controller by using multiple APDUs. It also contains the task of calculating only the Montgomery multiplication algorithm 7 on the security controller by sending multiple APDUs.
As mentioned in the Section 3.4.2 and shown in Figure 4.4 for an optimized authentication protocol a Montgomery multiplication algorithm on the security controller is computed by sending multiple APDU-Commands from an Android NFC-enabled smart phone to the security controller. Before any computation on the security controller side is applied, a reader initialization on the Android NFC-enabled smart phone occurs. The Montgomery multiplication algorithm on the reader is calculated for the randomly generated ephemeral key μ and a point x_P in the elliptic curve, by invocation of `ReaderInitialization(`μ,`x`$_P$`)` method provided by *MontgomeryMultiplication* library as shown in Figure 3.8. This function returns a transformed parameter of BigInteger data type (see Algorithm 6 Line 3) and for further processing on the security controller, it is necessary to convert this data to `byte[]`. This conversion is achieved using a function `BigInteger.tyByteArray()`, which returns a byte array containing the two's-compliment representation of this BigInteger. Infineon security controller provides various APDU-commands. General command is provided for getting the actual version of the security controller or as described in this master's thesis for a detection of the security controller. Furthermore, the following APDU-Commands for basic crypto field arithmetic operations are provided by the Inifineon security controller that are specially implemented for this project:

- Add - addition
- Div - division
- ModExp - modulation exponentation
- ModAdd - modulation addition
- Reduce - reduction
- ModInv - modulation inversion
- ModMult - modulation multiplication
- ModSub - modulation substraction
- Mult - multiplication

- Sub - substraction

However, not all of the above APDU-commands are required for an optimized ECC-based authentication protocol, but only those operations are taken into account that are necessary for Montgomery multiplication algorithm 7.

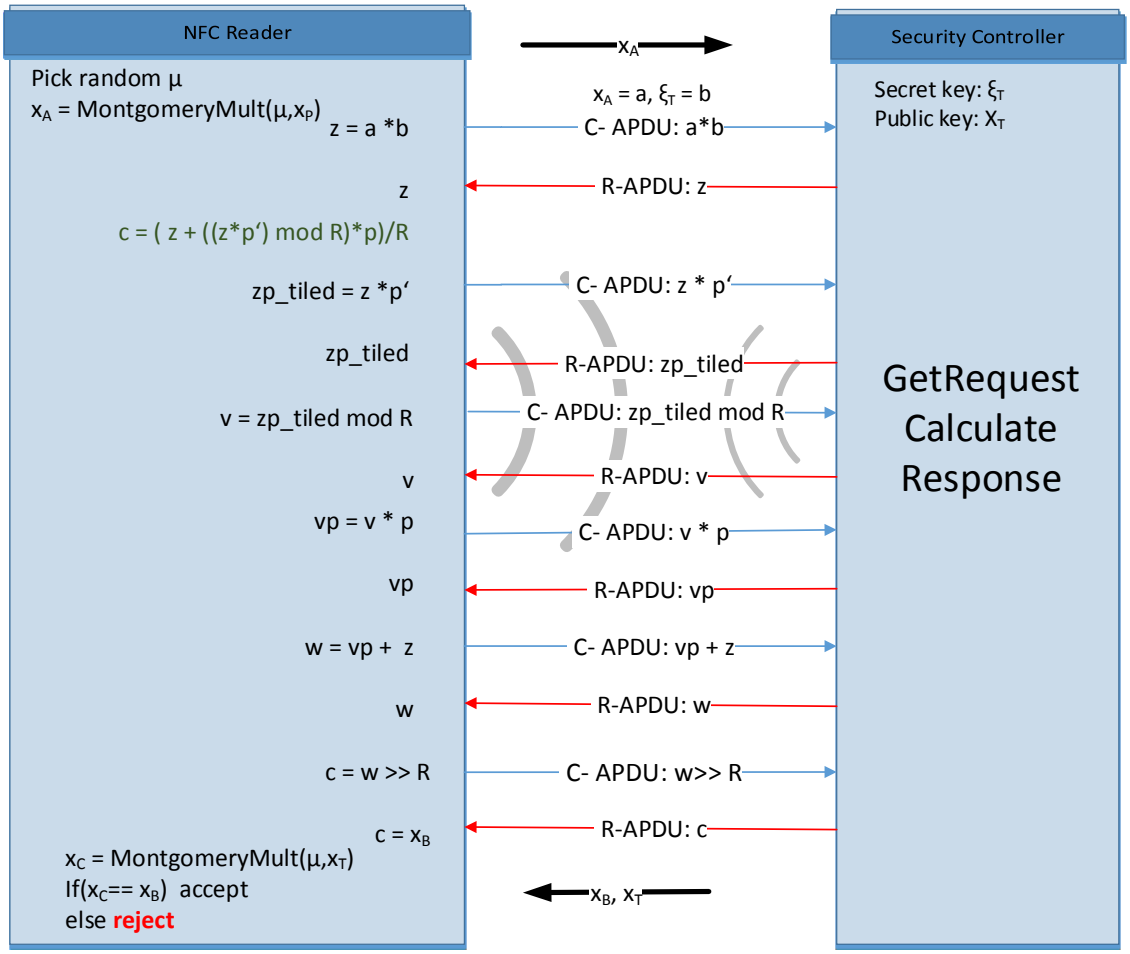


Figure 4.4: ECC-based authentication protocol by using many APDUs

Before any interaction occurrence between an Android NFC-enabled smart phone and the security controller, an initialization of a user interface is performed by *UserApplication* package, which consists of two JAVA entities as shown in Figure 3.4.

MainUI is the main class, where the detection of the security controller occurs. As long as no security controller is detected, the *ECCAuthenticateUI* entity is not initialized and no further functionality is performed.
ECCAuthenticateUI provides Android widgets where the authentication protocol with multiple APDUs is provided.

As the first arithmetic operation for computation of the Montgomery multiplication algorithm on the security controller, is a multiplication of two operands. C-APDU for

multiplication field arithmetic operation must be sent over an NFC interface from an Android NFC-enabled smart phone to the security controller in a byte block.

A construction of C-APDU multiplication field arithmetic operation from Table 4.1 is implemented in the Listing 4.6, where the first two elements of the Class (0xC0) and Instruction (0xF4) must always be set by default. The third byte consists of API APDU e.g. Multiply, Add etc. Furthermore, the other parameters mentioned above in the C-APDU structure are appended to the command e.g. parameters, response length etc. However, before a construction of a C-APDU occurs, a connection to the security controller must be established. After a successful connection establishment, a C-APDU is sent to

Table 4.1: Structure of C-APDU Multiplication Basic Field

C0	F4	xx	00	01	xx xx	xx 00	xx
		APDU			Length1 and Length2 of the operands	Expected length of the R-APDU	Two Operands with Length1 and length2

the security controller with the help of Android class `nfc.tech`, which provides a method `transceive(byte[] data)`. This method sends a raw command (C-APDU) to the tag and receives the response (R-APDU) from the tag. In order to access the NFC hardware, the permission must be set in the *AndroidManifest* file of the project as follows:

```
1    <uses-permission android:name="android.permission.NFC" />
```

Listing 4.6: C-APDU

```
1
2  public byte[] CallCryptoOperation(byte call, byte[] value) {
3
4      byte[] response = new byte[] {};
5
6      try {
7        IsoDep tag = IsoDep.get(this.connectedTag);
8
9        // connect to the tag
10       // request anticol select
11       tag.connect();
12
13       // If the call addresses the toolbox
14       // the value is the capdu
15       if (call>0x04)
16       {
17         // get response from the tag
18         response = tag.transceive(value);
19       }
20       else
21       {
22       byte[] capdu = new byte[value.length + 5];
23       capdu[0] = (byte) 0xC0;
24       capdu[1] = (byte) 0xF4;
25       capdu[2] = call; // APDU e.g. Mult, Add, Div etc.
26       capdu[3] = (byte) 0x00;
```

```
27        capdu[4] = (byte) 0x01; //value.length;
28        System.arraycopy(value, 0, capdu, 5, value.length);
29        // get response from the tag
30        response = tag.transceive(capdu);
31        }
32        // close the connection to the tag
33        tag.close();
34      } catch (Exception e) {
35        Log.e(Exception, e.getLocalizedMessage());
36      }
37
38      return response;
39    }
```

The `connect()` method opens a connection to the tag, and it enables I/O operations to the tag. In the case of an error during a connection establishment or sending a C-APDU to the security controller, it is immediately thrown an exception and no data exchange between an Android NFC-enabled smart phone and security controller is achieved.
An exception is thrown due to the following reasons:

- if the tag leaves the field
- **IOException**: if there is an I/O failure, or this operation is canceled

The `close()` method is called after an **I/O** operation. However, this function must not be called from the main application thread.
Afterwards, if a connection is successfully established, the security controller receives a C-APDU from the Android NFC-enabled smart phone. As the first task of the security controller is to check whether the received API APDU exists on the security controller's side or not. If the security controller provides no such API APDU, then it responds to the Android NFC-enabled smart phone with any incorrect value. If there is a valid API APDU provided on the security controller, it extracts the operands from the APDU and calculates the field arithmetic operation between the operands. After a calculation, the security controller responds to the Android NFC-enabled smart phone with an R-APDU command, which must have the following structure as shown in Table 4.2:

Table 4.2: Structure of R-APDU Multiplication Basic Field

Length of the results	Results with length3	Status code 0x00 means all OK	Status code of the transponder 0x90 0x00 means all OK

The response result from the security controller must be now compared with an expected result which is simulated by software (see Section 4.3). If both of the results (simulated and sent from the security controller) are the same, it is accepted that software simulation for this field arithmetic operation is correct and the basic crypto operation received by the security controller is correct too. Considering the other field arithmetic operations, the same procedure must be followed. For every APDU-Command of the field arithmetic operation, different parameters must be appended to the command e.g. the length of the result to be responded must be set in the C-APDU. After computation of

the last field arithmetic operation of the Montgomery multiplication algorithm from the security controller, its response must be converted again from the `byte[]` to `BigInteger` by using the following method `ByteArrayToBigInteger(byte[])` provided by *ECCAuthenticationUI* JAVA entity. Figure 4.4 shows a step by step calculation of Montgomery multiplication algorithm 7 on the security controller, by using multiple APDUs. In the case of the condition not executing `if` (Algorithm 7,*Line 3*), the affine x-coordinate x_B (in the Figure 4.4 shown as r) is sent from the security controller to the Android NFC-enabled smart phone without being subtracted from p for verifying the authenticity of the tag otherwise it must be subtracted. The verification on the reader is possible by invocation of `ReaderVerificiation(`x_B`,` x_T`)` method provided by *ECCAuthenticationProtocol* entity. An affine x_C is further computed by Montgomery multiplication algorithm for the ephemeral key μ and the public key x_T sent from the security controller. If the affine x_C and the computed affine x_B from the Montgomery multiplication algorithm on the security controller are equal, it is an authentic tag, otherwise it is rejected.
The verification of an authenticity on the reader side is not done using *Spongy Castle* library but through manually implemented Montgomery multiplication algorithm.

4.5 Implementation of Authentication Protocol with Single APDU

As elaborated in Section 3.4.3 an optimized ECC-based authentication protocol is achieved with the help of a single APDU. This means that the Android NFC-enabled smart phone sends a single C-APDU and receives a single R-APDU from the security controller in byte block as shown in Figure 4.5.
This section also elaborates on the computation of the Montgomery multiplication algorithm on the security controller with the help of a single APDU.

ECCAuthenticateUI entity class, part of *UserApplication* package shown in Figure 3.4 provides various Android *widgets* (Buttons), and one of these widgets is dedicated to achieve the authentication protocol with the help of a single APDU. The C-APDU construction of an authentication protocol with a single APDU is nearly the same as in Table 4.1, but differs from the construction procedure of C-APDU for field arithmetic operations. In this case the second operand and its length are not appended to the C-APDU, only a single parameter, an affine x-coordinate x_A is appended to the C-APDU. *TagCommunicator* entity class is again used to access the low level of NFC chip, and the same method Listing 4.6 is applied for data exchange between an Android NFC-enabled smart phone and the security controller. The flow procedure of receiving a C-APDU on the security controller side is the same as elaborated in the Sections 3.4.2 and 4.4. However, for this scenario on the security controller the Montgomery multiplication algorithm 7 is implemented and performs a single point multiplication. Converting the response value of the reader initialization to byte array, is the same as elaborated in the Section 4.4. After a computation of the Montgomery multiplication on the security controller's side, the security controller responds with a result saved in byte block (R-APDU) to the Android NFC-enabled smart phone. The structure for the construction of R-APDU remains the same as shown above in the Table 4.2. Converting the response of the security controller (from `byte[]` to `BigInteger`) for verifying the authenticity remains the same as in the

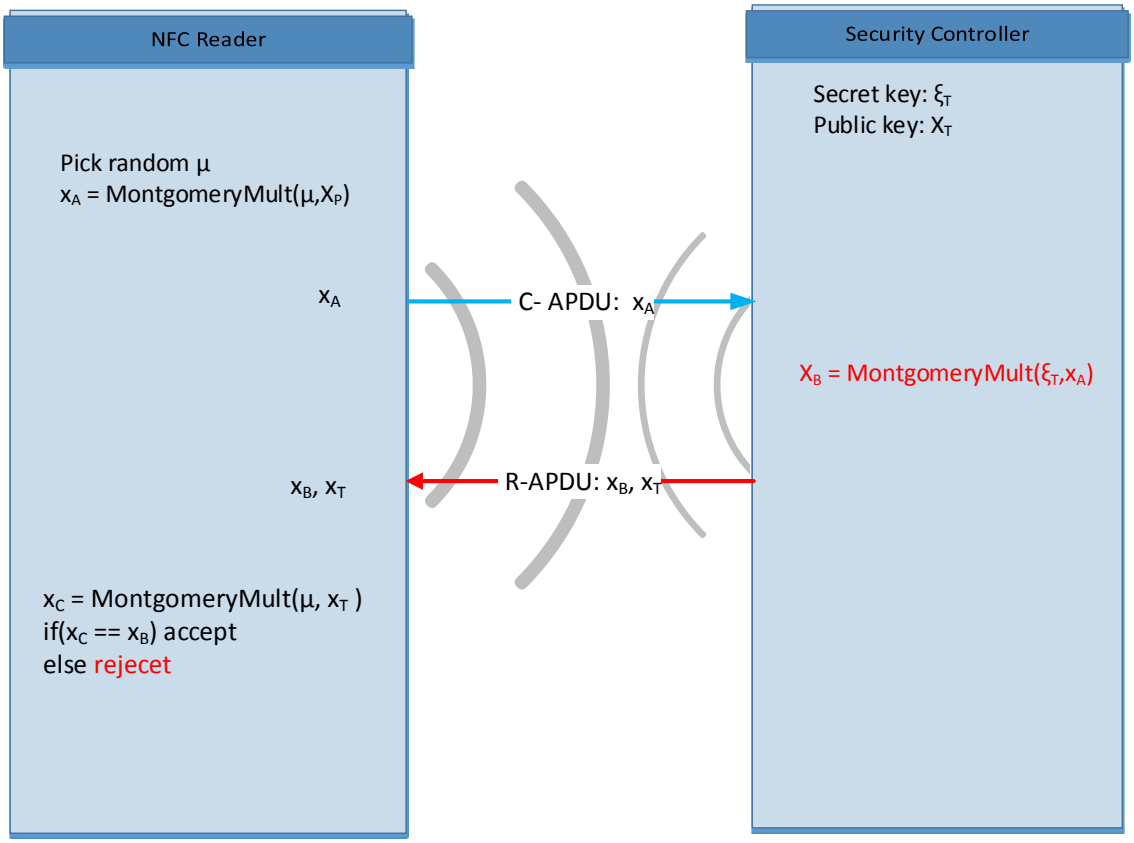


Figure 4.5: ECC-based authentication protocol by using a single APDU

Section 4.4. The affine x_C is computed using the Montgomery multiplication algorithm on the reader side, which is further compared with the affine x_B sent from the security controller. If both of them are the same, this is an authentic security controller and it is accepted, otherwise the security controller is rejected.

On the other side the correctness of an authentication protocol with a single APDU can also be further compared with the results obtained by the software simulation. If both of them are the same, it can be deduced that the security controller does a correct point multiplication, Montgomery multiplication algorithm 7 works properly and an optimized ECC-based authentication protocol is achieved with the help of a single APDU.

Chapter 5

Results

This chapter gives an overview of the results obtained during the implementation of an optimized ECC-based authentication protocol for security controller-based applications. The aim of this master's thesis is to achieve an efficient implementation of an optimized authentication protocol. The performance behavior of an NFC-enabled device and the security controller is then analysed. For a higher security aspect the processing time of data also increases, for the purpose of an authentication protocol. There have been various RFID tag authentication protocols delivered in the past. Each of them with their own advantages and drawbacks. In this master's thesis for the purposes of an optimized ECC-based authentication protocol a Montgomery multiplication algorithm defined over prime fields is consciously applied. The reason for applying an algorithm defined over prime fields is that this algorithm is particularly suitable for resource constrained embedded systems, furthermore, Infineon security controller also supports SLE multiplier. This chapter gives the results of the software simulation, for arithmetic operations, Montgomery multiplication algorithm and authentication protocol for different elliptic curves defined over prime field, and binary field.
This chapter further describes timing analysis of an authentication protocol for $secp192r1$ elliptic curve defined over prime field, in the interaction between an NFC-enabled smart phone, and the security controller.

5.1 Android Simulator

An optimized ECC-based authentication protocol is simulated with a simple application developed for Android OS. The connection establishment, data exchange between an NFC-enabled smart phone and the security controller (necessary Libraries invocation) are described in chapters 3 and 4. Whenever a user starts the application, the main user interface is displayed as shown in Figure (a) 5.1. As long as no security controller is detected that is not compatible with our application, the next Figure (b) 5.1 will not be displayed to the user. In the case of the security controller detection, the version of the security controller is immediately retrieved, and displayed in Figure (b). The other buttons displayed in Figure (b) are for different purposes, although each of them is suited for an authentication protocol.

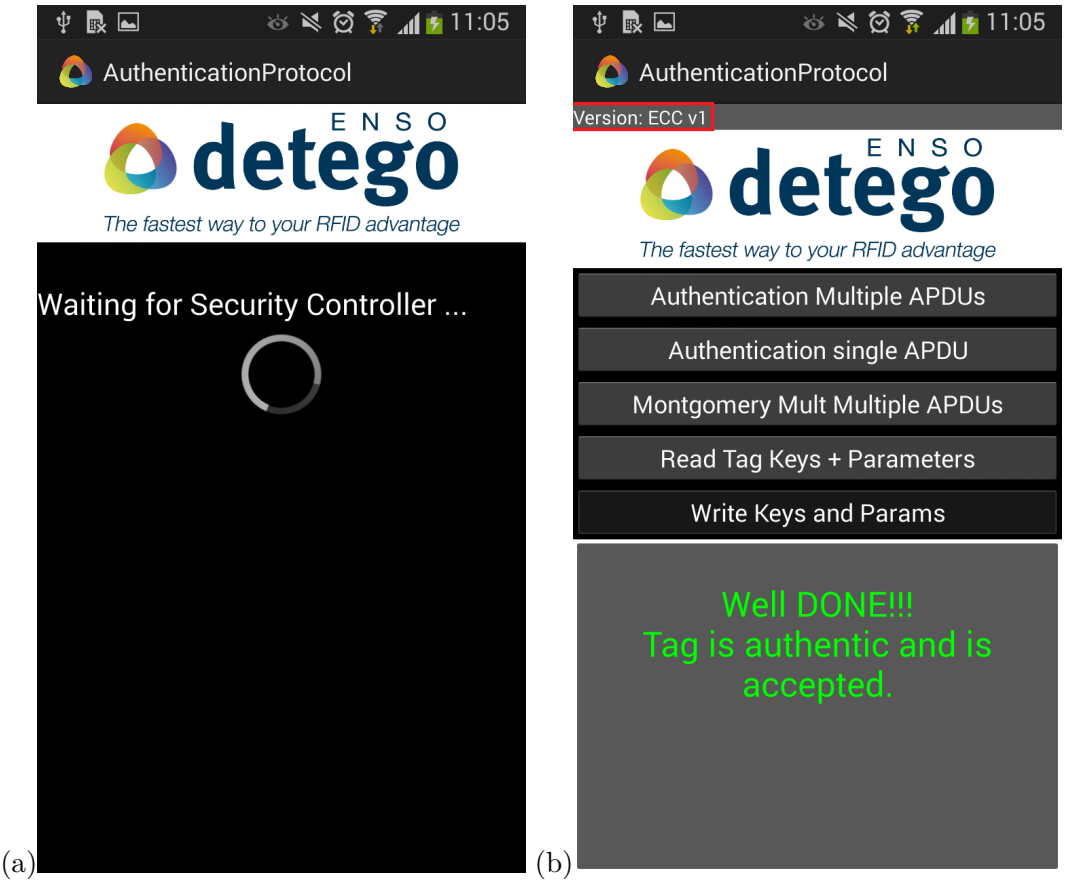


Figure 5.1: (a) Main User Interface, (b) Interaction User Interface

For example, **Montgomery Mult Multiple APDUs** button as shown in Figure (b) 5.1 performs the computation of Montgomery multiplication algorithm on the security controller by sending multiple APDUs from an Android NFC-enabled smart phone to the security controller. Each of the APDUs consists of field arithmetic operations required by Montgomery multiplication algorithm.

As shown in Figure (b) 5.1 the **Authentication Multiple APDUs** button is a continuation work of the above option, furthermore, only the reader initialization and the verification of the authenticity of the protocol for the security controller is implemented on an Android NFC-enabled smart phone. Single APDU interaction between an Android NFC-enabled smart phone and the security controller is performed by using the **Authentication single APDU** button, where Montgomery multiplication algorithm is computed on the security controller with the help of a single request sent from an NFC-enabled smart phone. In the case of the authentication protocol with single APDU, as mentioned above, only an initialization and verification of the authenticity is included. So for both of the options (Multiple vs Single APDUs), if the the tag is authentic the message **WELL DONE!!! Tag is authentic and is accepted** is displayed, otherwise **Tag is rejected** is displayed to the user.

5.2 Software Simulation

Before applying an authentication protocol between an NFC-reader and the security controller, a software simulation must be analyzed, designed and implemented which presents a real authentication protocol. The results obtained during the software simulations can further help a real authentication between NFC-reader and the security controller. It is important to mention that during the simulations we decided to also realize a software simulation for the other elliptic curves defined over prime field e.g. *secp192r1, secp224r1, secp521r1* and *sect163r1* elliptic curve defined over binary field. Although considering the sections 5.3 and 5.4, for an optimized ECC-based authentication protocol between NFC-reader and the security controller interaction, only the *secp192r1* elliptic curve defined over prime field is applied.

5.2.1 Montgomery multiplication over prime field

As elaborated in the design 3 and the implementation 4 chapter, for the purpose of an authentication protocol only a single point multiplication is applied on the security controller, which is very suitable for NFC-enabled embedded devices and shows an efficient performance. We were not only interested in achieving an efficient authentication protocol, which will be suitable for the security controller authentication, but also to achieve an efficient processing time that is suitable for NFC-reader constraints e.g. battery life. Figure 5.2 shows a software simulation timing analysis of a Montgomery multiplication algorithm 7 for elliptic curves defined over prime field, which is simulated on an Android NFC-enabled smart phone. The column *Round* in the Table 5.1 shows the execution time of the first round or known as cycle of any operation in the software simulation. The first round almost always requires longer execution time than the other first 10 rounds, or longer than the second round. This is due to the first miss in the cache, the processor then fetches the data from main memory.
Montgomery multiplication algorithm for *secp192r1* elliptic curve, for the first cycle requires a slightly shorter processing time than *secp224r1* elliptic curve defined over prime field. On the other side, *secp521r1* elliptic curve takes twice as long as the other elliptic curves with the shorter key length. The minimum execution time of a Montgomery multiplication algorithm for the *secp192r1* elliptic curve in a software simulation is achieved within $90.62\,\mu s$, where the maximum execution time is also influenced by JAVA OOP, and it is estimated in an approximate execution time of $281\,\mu s$. An approximate statistical average execution time of a Montgomery multiplication algorithm is achieved in a time of $129.37\,\mu s$.
Montgomery multiplication algorithm for the elliptic curve *secp224r1* is simulated in a minimum time of $92.08\,\mu s$. The maximum necessary execution time of a Montgomery multiplication algorithm for the *secp224r1* elliptic curve is $311.33\,\mu s$, where an average execution time of the algorithm is $131.2\,\mu s$.

The elliptic curve *secp521r1* needs more time for its software simulation than the above elliptic curves. The minimum execution time for a Montgomery multiplication Algorithm of 521-bit elliptic curve is achieved within $124.21\,\mu s$. The maximum simulation time differs also from the 192-bit and 224-bit elliptic curves and is achieved in approximately

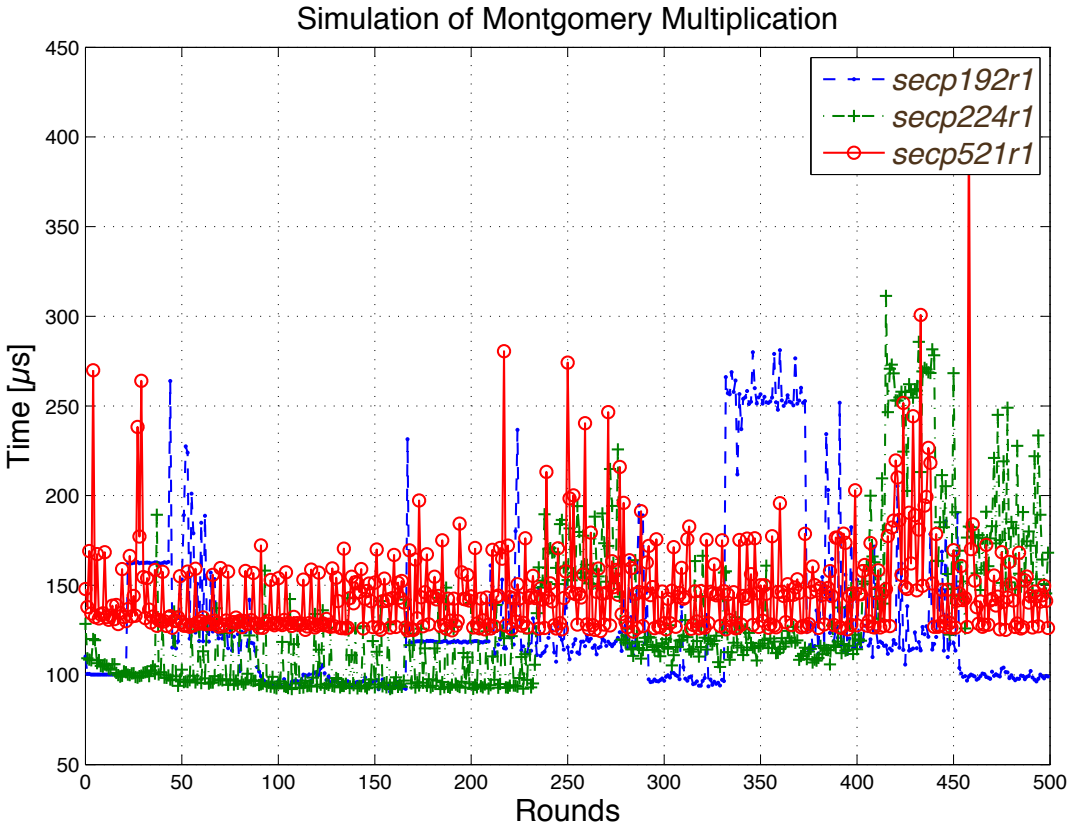


Figure 5.2: Software Simulation: Montgomery Multiplication for Prime Field

762.71 μs. A statistical average for Montgomery multiplication algorithm of 521-bit in a software simulation is achieved within 146.43 μs

The summary of software simulation for Montgomery multiplication algorithm captured from Figure 5.2 is shown in Table 5.1.

Table 5.1: Software Simulation of a Montgomery Multiplication

Execution time	*First Round*	*Minimum*	*Maximum*	*Average*
secp192r1	110.21 μs	90.62 μs	281.12 μs	129.37 μs
secp224r1	128.46 μs	92.08 μs	311.33 μs	131.2 μs
secp521r1	147.88 μs	124.21 μs	762.71 μs	146.43 μs

It is very important to mention that during the software simulation a Garbage Collector (GC) is thrown from an Android NFC-enabled smart phone OS. **GC_FOR_ALLOC** means that the GC was triggered because there was not enough memory left on the heap to perform an allocation. It might also be triggered when new objects are created. This also affected the timing analysis where a single multiplication sometimes needed an extremely long processing time. In order to show the time required for Montgomery multiplication in detail this means being able to see the minimum required processing time, the time needed for these steps are filtered. Otherwise if these cycles were not filtered where a GC

appeared, the performance of timing analysis was illegible/unreadable. However, for nearly every test of a timing analysis of any operation e.g. Multiplication, Addition, Division etc. a GC appeared. Montgomery multiplication algorithm for the elliptic curves (*secp192r1*, *secp224r1*, *secp521r1*) defined over prime field shows an efficient performance and requires a shorter processing time, in comparison to Montgomery multiplication algorithm for elliptic curves defined over binary field (see Section 5.2.2).

5.2.2 Montgomery Algorithm over Binary Field

Software simulation of a Montgomery multiplication algorithm defined over prime field for the purpose of one-way authentication protocol is faster than the algorithm defined over binary field. The source code of a Montgomery multiplication algorithm for the elliptic curve

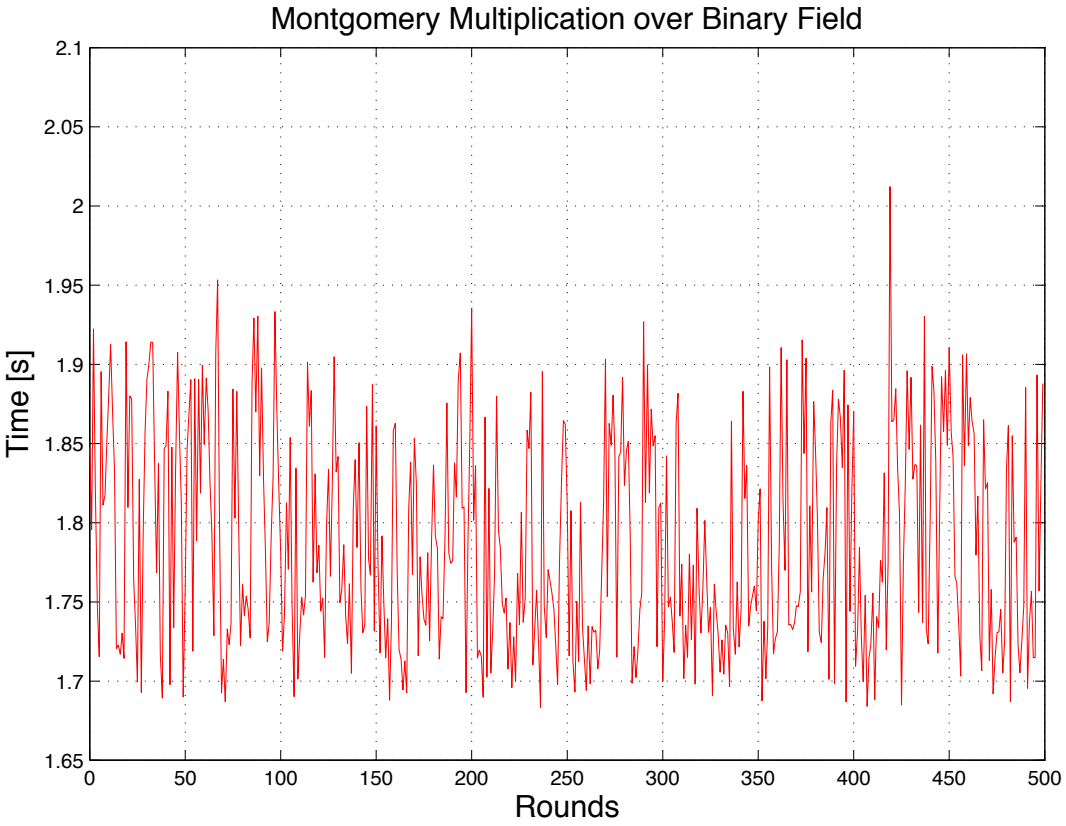


Figure 5.3: Software Simulation: Montgomery Multiplication over F_{2^p}

defined over binary field is taken with permission from Höller's Master Thesis [Höl13], and further adapted to simulate an Android NFC-enabled smart phone. As an elliptic curve defined over binary field for one way authentication protocol, is a 163-bit elliptic *sect163r1* chosen and this is simulated by the invocation of `montgomery_mult(FIELD2N,FIELD2N, FIELD2N)` method provided by the *elliptic_poly* entity.

The minimum processing time required by the NFC-enabled smart phone to simulate a Montgomery multiplication algorithm 1 defined over binary field is 1.68 s, and a maximum time of 2.01 s, whereas an average processing time of a scalar multiplication is 1.79 s. The

first cycle of Montgomery Multiplication Algorithm defined over binary field simulated in software requires a time of 1.85 s.

5.2.3 Field Arithmetic Operations

A comparison of field arithmetic operations of the elliptic curves defined over prime field is necessary. In both of the above chapters (design and implementation), it was elaborated that as the most expensive field arithmetic operations of elliptic curve are division, multiplication and reduction (modulation) arithmetic operations. Whereas, the division arithmetic operation for Montgomery multiplication algorithm defined over prime field is achieved by applying shifting operation.

5.2.3.1 Addition

Software simulation of the addition field arithmetic operation for elliptic curves defined over prime field is shown in Figure 5.4. Even from Figure 5.4 the difference in processing

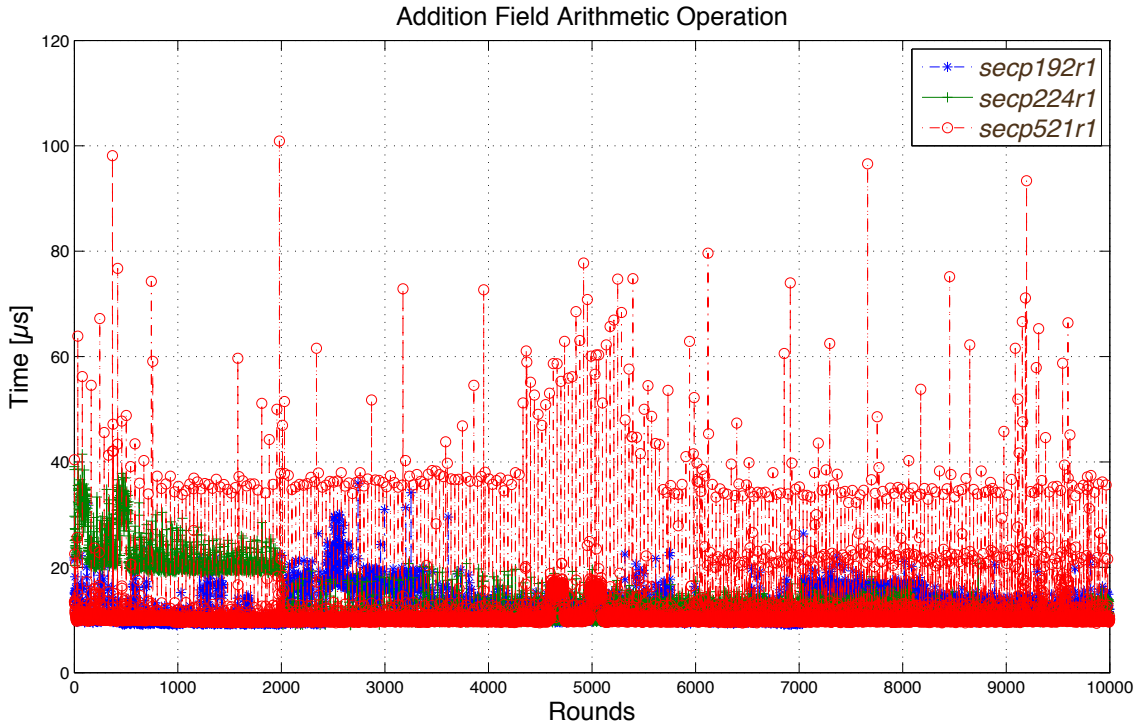


Figure 5.4: Software Simulation: Addition Field Arithmetic Operations over F_p

time cannot be clearly recognised between *secp192r1* and *secp224r1* elliptic curves, it shows that 521-bit elliptic curve (*secp521r1*) needs a longer processing time for the addition field arithmetic operation. A summary of minimum, maximum and average processing time for the addition field arithmetic operation of the elliptic curves defined prime field captured from Figure 5.4 is shown in Table 5.2.

Table 5.2: Software Simulation of Addition Operation

Execution time	*First Round*	*Minimum*	*Maximum*	*Average*
secp192r1	21.79 μs	8.96 μs	35.95 μs	9.63 μs
secp224r1	39.12 μs	9 μs	41.58 μs	11.92 μs
secp521r1	40.42 μs	9.37 μs	100.91 μs	13.62 μs

5.2.3.2 Reduction

Multiplication and division field arithmetic operations are considered as very expensive operations. Although there is another more expensive operation which requires a longer

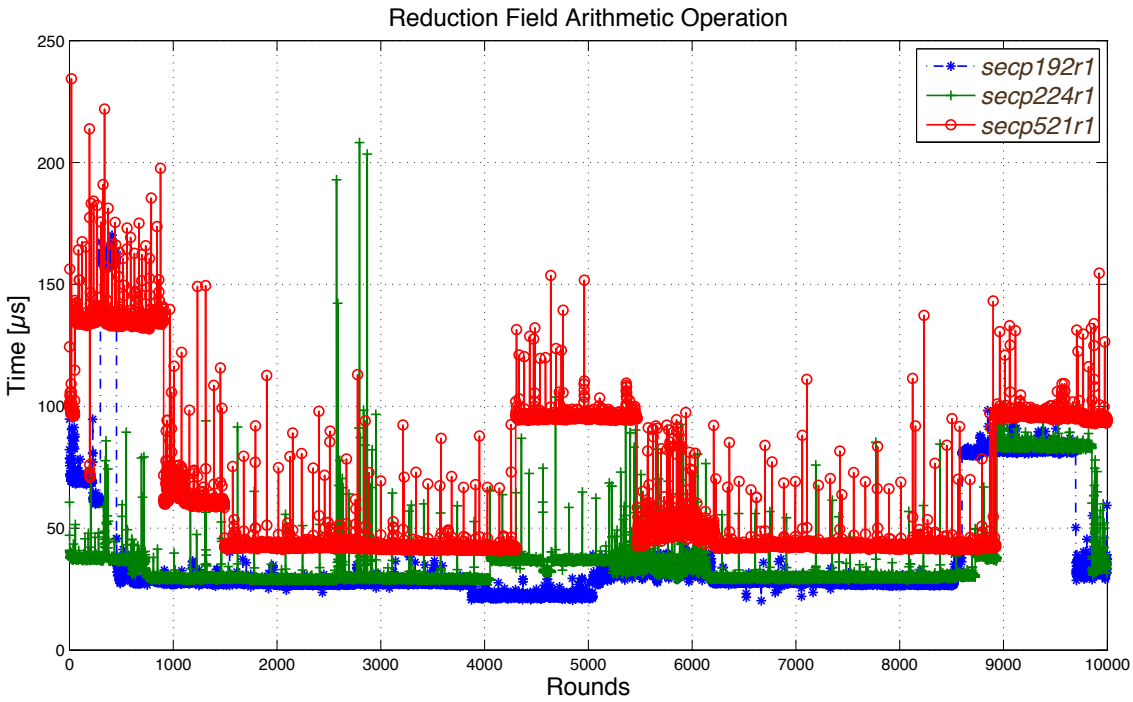


Figure 5.5: Software Simulation: Reduction Field Arithmetic Operations over F_p

processing time than these operations, and this known as reduction or modulation field operation. As shown in Figure 5.5 the modulation field arithmetic operation for a 512-bit elliptic curves takes a longer data processing time than 192- and 224-bit elliptic curves. Modular reduction is a simple masking to b bits.

A summary of timing analysis for three different elliptic curves defined over prime field

Table 5.3: Software Simulation of Modulation Operation

Execution time	*First Round*	*Minimum*	*Maximum*	*Average*
secp192r1	94.75 μs	20.25 μs	170.79 μs	37.38 μs
secp224r1	60.67 μs	28.33 μs	208.17 μs	37.45 μs
secp521r1	124.42 μs	40.67 μs	234.37 μs	65.31 μs

captured from Figure 5.5 is shown in Table 5.3, where *secp521r1* elliptic curve with 512-bit

key length differs from the other elliptic curves with the shorter key length.

5.2.3.3 Multiplication

Montgomery multiplication algorithm consists of various arithmetic operations, and a software simulation of each of them is analysed. Figure 5.6 shows the simulation of multiplication field arithmetic operation of various length for *secp192r1, secp224r1, secp521r1* elliptic curves.

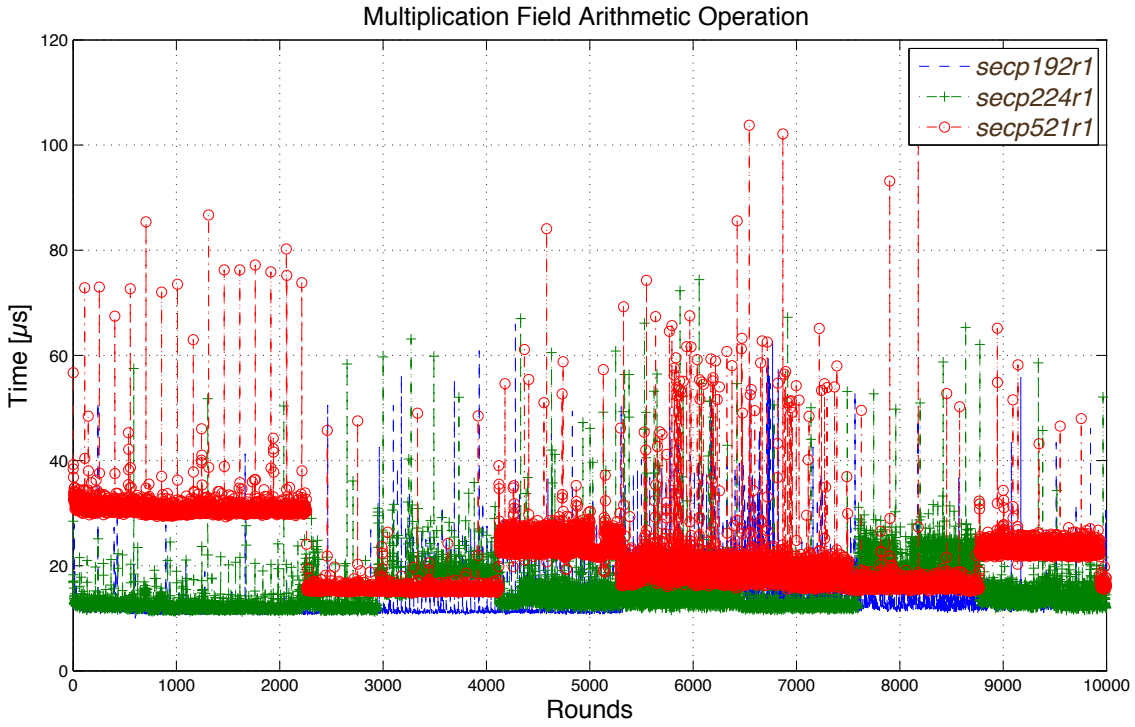


Figure 5.6: Software Simulation: Multiplication Field Arithmetic Operations over F_p

In comparison to the reduction field arithmetic operation (see Section 5.2.3.2) as one of the most expensive operation in cryptography, multiplication field arithmetic operation shows a better performance in timing analysis. The summary for different lengths of operands, a multiplication timing analysis for three elliptic curves defined over prime field captured form the Figure 5.6 is shown in Table 5.4.

Table 5.4: Software Simulation of Multiplication Field Arithmetic Operation

Execution time	*First Round*	*Minimum*	*Maximum*	*Average*
secp192r1	26.75 μs	10.04 μs	65.96 μs	13.58 μs
secp224r1	28.46 μs	11.2 μs	74.42 μs	15.18 μs
secp521r1	56.71 μs	14.79 μs	111.46 μs	22.37 μs

5.2.3.4 Right Shift Operator instead of Division

The core of Montgomery multiplication algorithm is to avoid the division arithmetic operation. For basics of mathematics related to the Montgomery multiplication algorithm over prime field, see Section 2.4.3.2. The division arithmetic operation in the algorithm is achieved by applying a **right-shift** by b bits, which is very efficient if b is a multiple of the word bitsize. A summary of right-shifting operation timing analysis for three elliptic curves defined over prime field captured from Figure 5.7 is shown in Table 5.5.

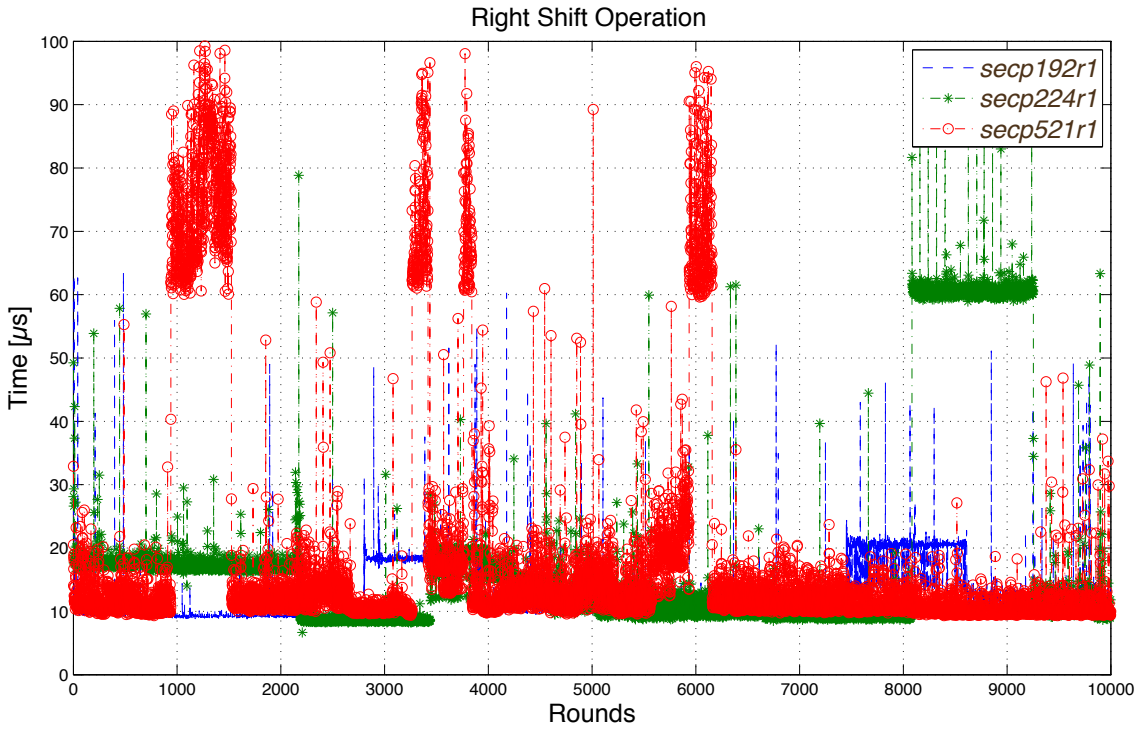


Figure 5.7: Software Simulation: Right Shift Operation over F_p

In comparison to reduction operation the difference between other field arithmetic operations such as addition, multiplication and right-shifting is very low.

Table 5.5: Software Simulation of Right Shift Operation

Execution time	*First Round*	*Minimum*	*Maximum*	*Average*
secp192r1	26.37 μs	8.79 μs	63.54 μs	12.44 μs
secp224r1	49.25 μs	6.67 μs	94.96 μs	18.50 μs
secp521r1	32.97 μs	9.33 μs	99.25 μs	18.95 μs

5.2.4 Authentication Protocol over prime field

A timing analysis of a software simulation for an optimized ECC-based one-way authentication protocol is as follows.

For one-way authentication protocol of the security controller only a single multiplication is applied on the security controller. The input parameters of a Montgomery multiplication algorithm must be transformed as explained in the Design Chapter 3. These transformations are very expensive operations for the elliptic curves, therefore the transformation of the operands in this master's thesis is also achieved by using Montgomery multiplication algorithm 7. The parameters that must be transformed on the reader's side are: randomly generated ephemeral key μ, and a point x_P on a elliptic curve. The security controller's parameters that must be transformed are: secret key ξ_T and the public key x_T which is the affine x-coordinate.

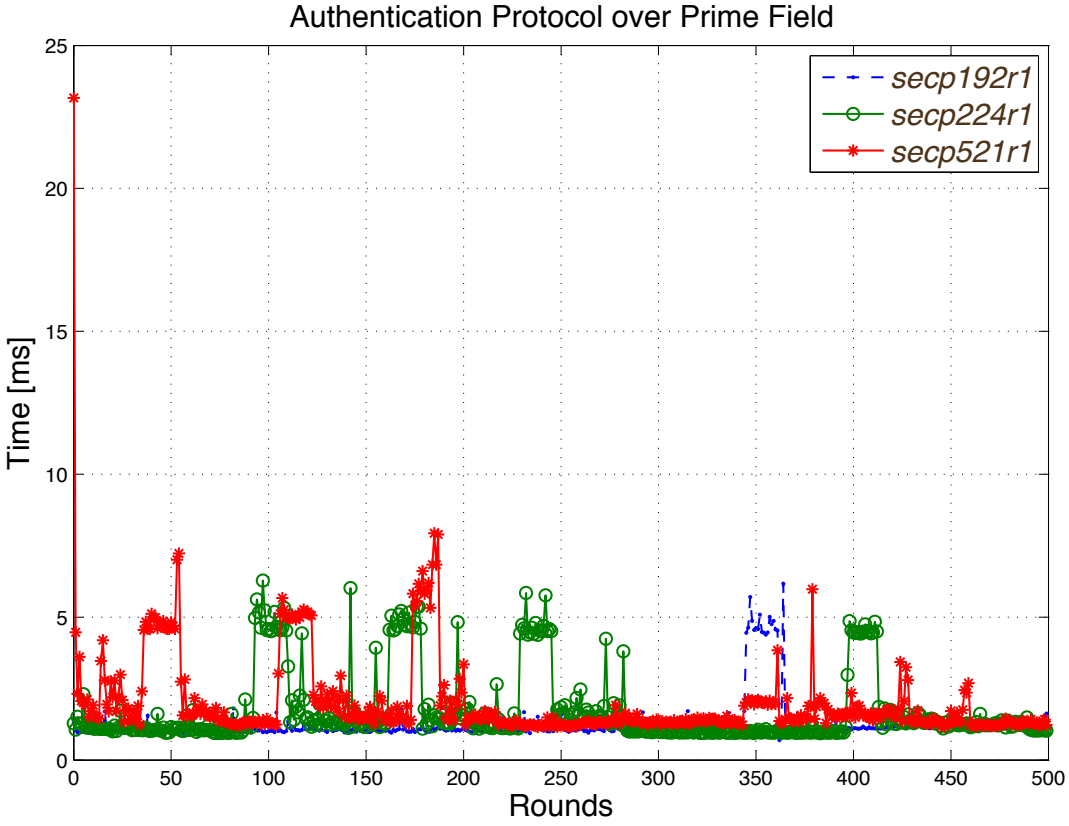


Figure 5.8: Simulation of Authentication Protocol over Elliptic curves defined over F_p

Software simulation of one-way authentication protocol for the elliptic curves defined over prime field is applied whereby the transformation of the operands as elaborated in Section 3.3 is included within the simulation of timing analysis, and it is shown in Figure 5.8.

A summary of software simulation of one-way authentication protocol for elliptic curves defined over prime field, where the transformation of parameters $(\mu, x_P.\xi_T, x_T)$ part of a protocol, is achieved by using Algorithm 7, captured from Figure 5.8 is shown in Table 5.6

Table 5.6: Software Simulation of Authentication Protocol

Execution time	*First Round*	*Minimum*	*Maximum*	*Average*
secp192r1	1.3 *ms*	0.69 *ms*	6.17 *ms*	1.24 *ms*
secp224r1	1.28 *ms*	0.94 *ms*	6.28 *ms*	1.74 *ms*
secp521r1	23.16 *ms*	1.15 *ms*	23.16 *ms*	2.02 *ms*

5.2.5 Authentication Protocol over binary field

The source code of an authentication protocol for the elliptic curve defined over binary field is taken with permission from Höller's Master Thesis [Höl13], and further adapted to simulate an Android NFC-enabled smart phone. The software simulation of one-way authentication protocol for *sect163r1* elliptic curve defined over binary field takes a longer processing time than elliptic curves defined over prime field as shown in the Figure 5.9. In Section 5.2.2, the timing analysis of a Montgomery multiplication over binary field is computed without including the calculation of the public key, constructing an elliptic curve, computing a point within this elliptic curve etc.

As Höller [Höl13] elaborated that the source code of the field arithmetic functions (multi-

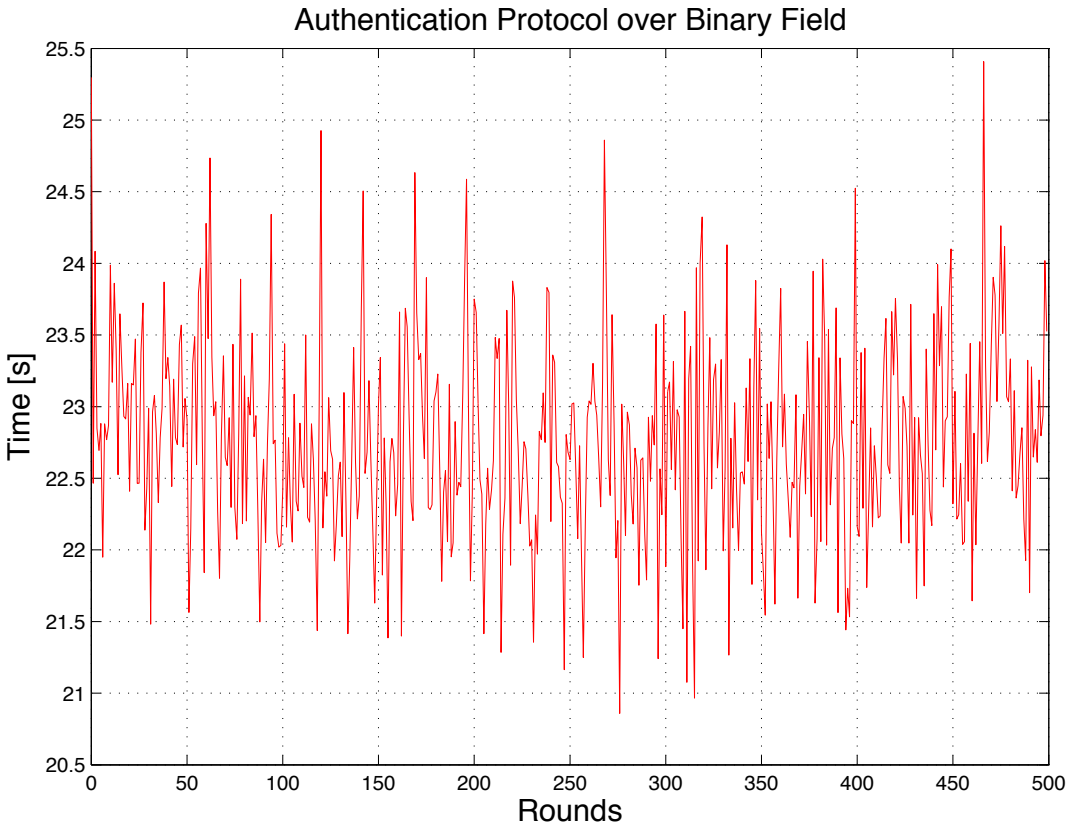


Figure 5.9: Simulation of Authentication Protocol for elliptic curves defined over F_{2^p}

plication, addition, division, subtraction, squaring) for software-based ECC is taken from [Ros], although for one-way authentication protocol an adaption was necessary. Therefore

an authentication protocol takes a relatively longer processing time than a single point multiplication, due to the challenge computation. The minimum processing time for an authentication protocol for *sect1631r1* curve defined over binary field is 20.86 s, a maximum time is 25.41 s, and a statistical average execution time is 22.76 s. The first cycle of an authentication protocol for binary field simulated in a software requires 25.30 s. It is interesting to see how the authentication protocol takes so much time in comparison to Montgomery multiplication algorithm defined over binary field (see Section 5.2.2), this is due to elliptic point multiplication. An elliptic point multiplication for binary field takes nearly 8 times longer then Montgomery multiplication Algorithm defined over binary field. The elliptic point multiplication is achieved by the invocation of `poly_elptic_mul(FIELD2N, POINT, CURVE)` provided by *elliptic_poly* entity field. We have to compute a public key which is a point multiplication of a private key (an integer to base 2, not normal basis), and a point on the elliptic curve defined over binary field [Kob92]. Also the affine x-coordinate of challenge x_A is an elliptic point multiplication. This means that the timing of an authentication protocol is much impacted from the point multiplication in our test cases.

It is very important to mention that parameters are defined as **short** type, and each of the arithmetic field operations defined over binary field (see book [MvOV01]) is implemented manually.

5.3 Multiple APDUs Interaction

Software simulation mimics a simulation of an authentication protocol, which is further applied between an Android NFC-enabled smart phone and the security controller. As mentioned and elaborated in the design chapter 3, the interaction between an NFC-reader and the security controller is initiated over an NFC interface by using APDU, where the message is saved in a byte block. This performs a real communication between an NFC-reader and the security controller. Chapter 4 shows that a family of Infineon security controller provides various APDUs for different purposes. In order to compute a Montgomery multiplication (Algorithm 7), each of its arithmetic operations are computed by sending a C-APDU separately over an NFC-interface saved in APDUs from Android NFC-enabled smart phone to the security controller. The security controller must respond to the Android NFC-enabled smart phone for each of C-APDUs, with an APDU known as R-APDU. Futhermore, an optimized authentication protocol is also realised by using multiple APDUs, with an integration of reader initialization, and the verification of an authenticity on the smart phone's side.

5.3.1 Montgomery multiplication over prime field

The same procedure of timing analysis applied in the software simulation is further applied in this section. The first step to achieve an optimized ECC-based authentication protocol is a computation of a single point multiplication on the security controller. As mentioned above the Montgomery multiplication algorithm is implemented and computed by requiring a computation for each of its field arithmetic operations from the Infineon security controller. In order to compute the requests sent from the Android NFC-enabled

smart phone, Infineon security controller responds with an R-APDU separately as shown in the Figure 4.4.

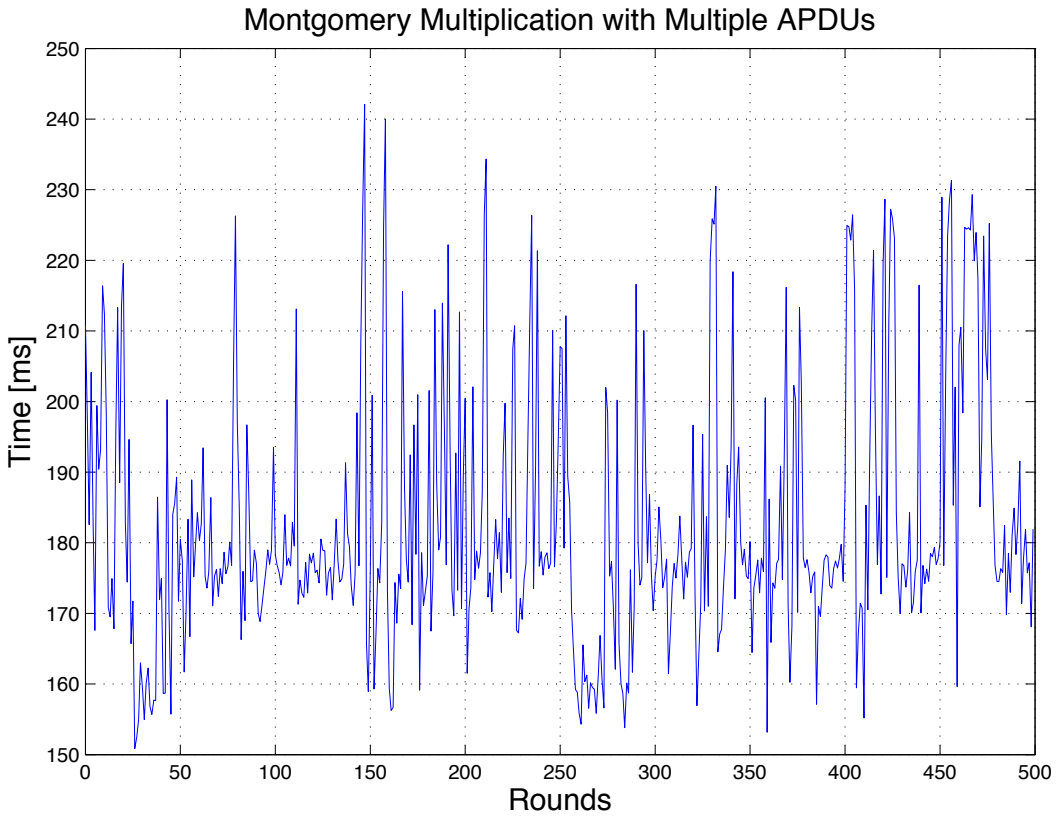


Figure 5.10: Security Controller Response: Montgomery Multiplication over *secp192r1*

Time is measured through taken into account the time of the operating system running on an Android NFC-enabled smart phone. Figure 5.10 shows a timing analysis of a Montgomery multiplication algorithm defined over prime field, where for each of its field arithmetic operations a C-APDU is sent from the Android NFC-enabled smart phone to the security controller. The minimum execution time required to compute a Montgomery multiplication Algorithm for elliptic curve *secp192r1* by sending multiple APDUs is 150.85 ms. The time required to compute a Montgomery multiplication with multiple APDUs is not constant, it varies depending on the other processes executed in the background by the Android NFC-enabled smart phone, but also an the connection establishment between NFC-enabled smart phone and the security controller. As shown in the Figure 5.10, there are some rounds which need a longer execution time and the maximum execution time required to compute a Montgomery multiplication algorithm with multiple APDUs within 500 rounds is 242.07 ms.

Finally, the average execution time of a Montgomery multiplication algorithm where multiple C-APDUs are sent to the security controller and multiple R-APDUs are received, is achieved within 182.58 ms.

5.3.2 Authentication Protocol with Multiple APDUs

This section is as a continuation of the work of Section 5.3.1 where an optimized ECC-based authentication protocol for the security controller is also achieved with the help of multiple APDUs.

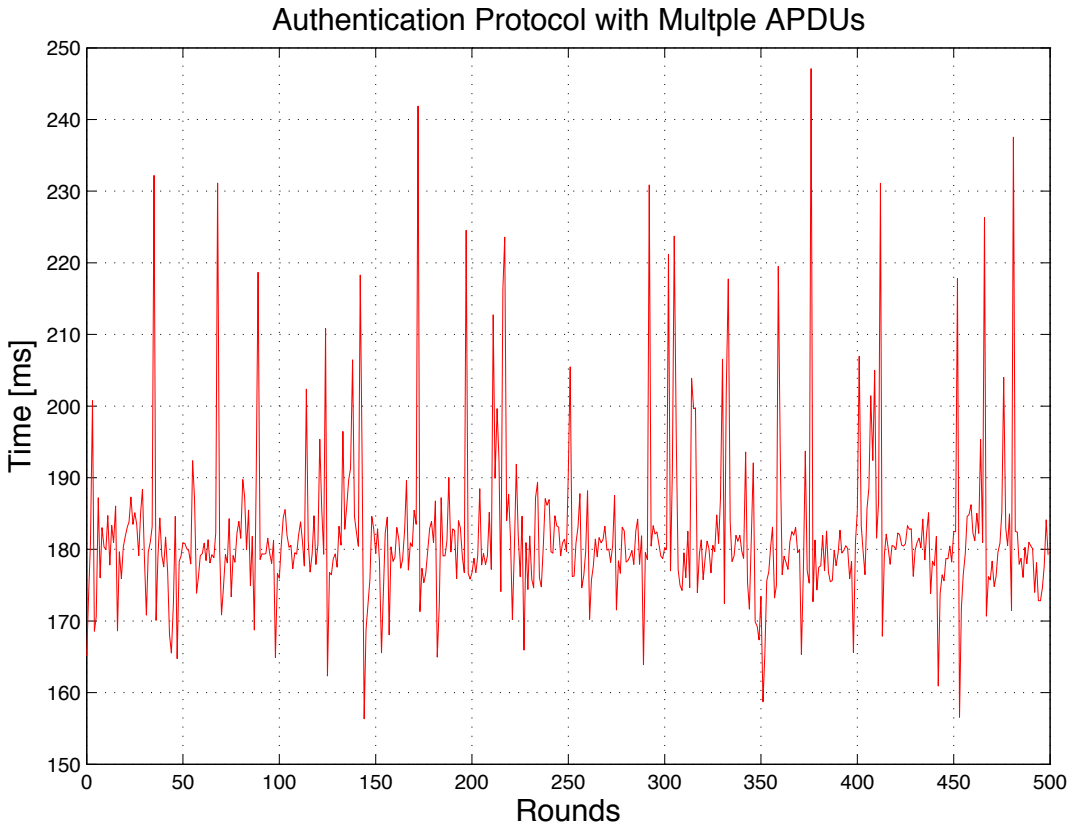


Figure 5.11: Security Controller Response: Protocol with Multiple APDUs over $secp192r1$

Figure 5.11 shows a timing analysis of an authentication protocol with multiple APDUs. On the security controller's side the private key ξ_T is saved on it and a challenge x_A is computed by Montgomery Multiplication Algorithm on reader side and then sent to the security controller.

On the reader's side, for an initialization and the verification of an optimized authentication protocol, a Montgomery multiplication algorithm is computed for the given parameters.

The minimum time for an optimized ECC-based authentication protocol with multiple APDUs interaction between an Android NFC-enabled smart phone and the security controller is 156.34 ms, and the maximum time is 247.09 ms. A statistical average for an optimized ECC-based authentication protocol by applying multiple APDUs between reader and security controller is achieved in an approximate time of 182.88 ms.

5.4 Single APDU Interaction

As shown in the Figure 4.5, a single APDU is sent from an Android NFC-enabled smart phone to Infineon security controller, for the purpose of an optimized ECC-based authentication protocol. This form of authentication protocol is feasible for the security controller (RFID tag).

5.4.1 Montgomery Multiplication with Single APDU

In the Section 5.3.1, the Montgomery multiplication algorithm on the security controller is computed with the help of multiple APDUs, whereas this section shows a computation of a Montgomery multiplication algorithm with the help of a single APDU.

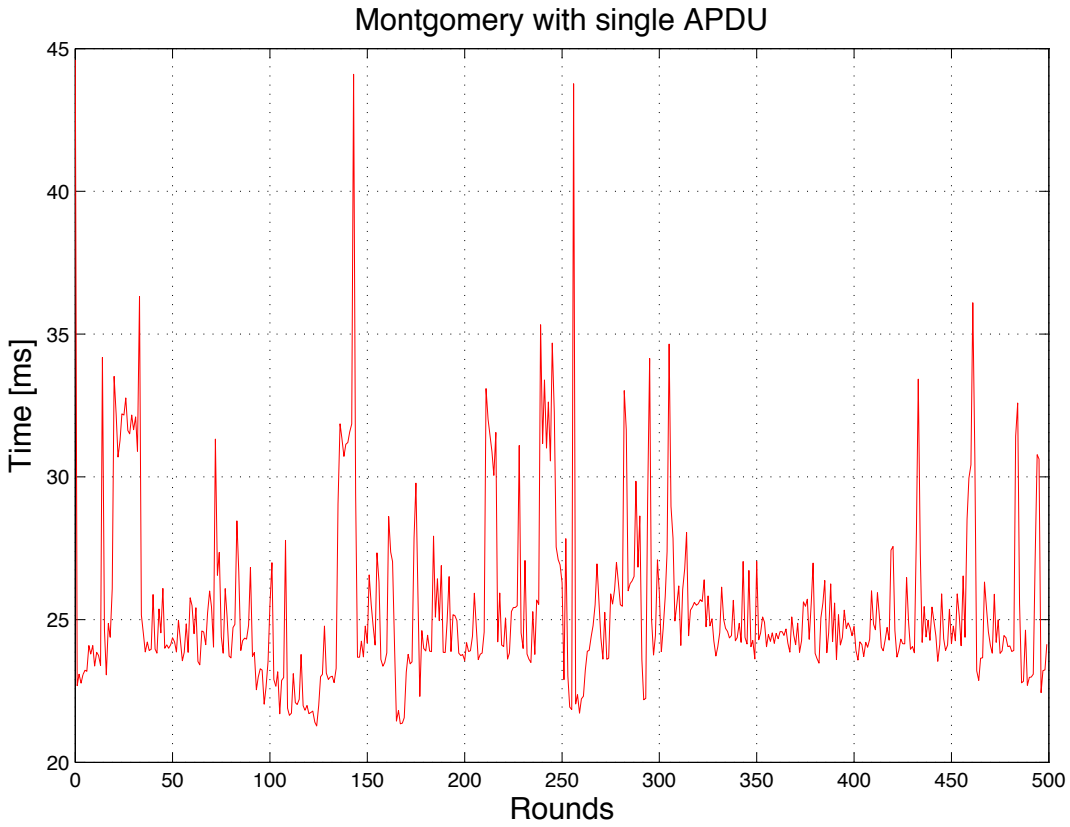


Figure 5.12: Security Controller Response: Montgomery Multiplication over *secp192r1*

Security controller's secret key is saved on it.

A challenge parameter x_A is sent from NFC-reader (see Section 5.4.2) to the security controller. The minimum execution time for a computation of a Montgomery multiplication algorithm on the security controller with a help of single APDU is achieved within 21.26 ms, as shown in the Figure 5.12, which is very optimal for an RFID tag authentication protocol.

The same as in Multiple APDUs Interaction Section 5.3, the execution time for a single

APDU varies depending on processes executed in background which impacts the speed of a computation of Montgomery multiplication algorithm and also the connection time to the security controller. The maximum execution time required by security controller to compute a Montgomery multiplication algorithm with a single APDU is 44.61 ms.

Finally, the average execution time of a Montgomery multiplication algorithm, where a single APDU response is sent to the security controller and a single is received is achieved within 26.02 ms.

5.4.2 Authentication Protocol with Single APDU

The aim of this master thesis is to achieve an optimized ECC-based authentication protocol with the help of a single APDU. This task is an extension of Section 5.4.1 and Figure 5.13 shows a timing analysis of an authentication protocol with a single APDU. Within this timing analysis is taken into account, a reader initialization (random ephemeral key, selecting a point on the curve E), multiplication on the security controller, and the reader verification to validate the authenticity of the tag. The verification of the authenticity is done on the Android NFC-enabled smart phone side.

The minimum time for an authentication with a single APDU is 22.03 ms, which is similar

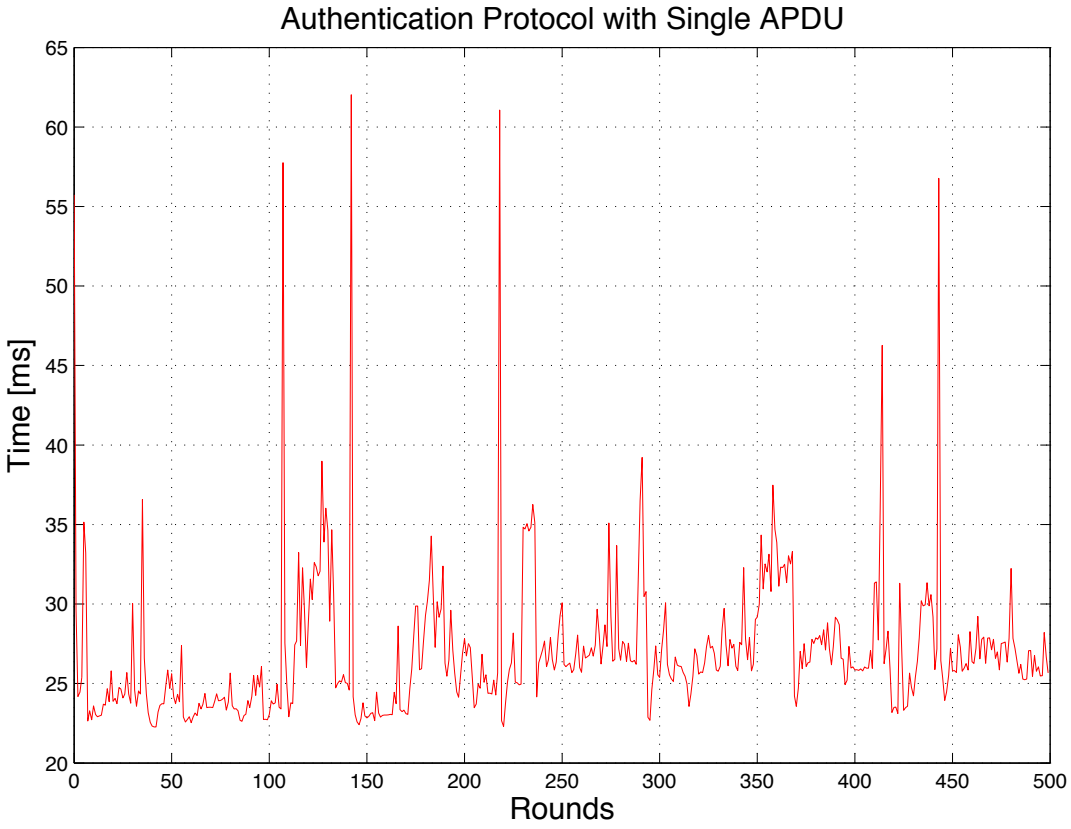


Figure 5.13: Security Controller Response: Authentication Protocol over $secp192r1$

to the processing time for a computation of a Montgomery multiplication Algorithm with a single APDU (see section 5.4.1). The maximum time for an optimized ECC-based

authentication protocol for an Infineon security controller with a help of a single APDU is 62.02 ms. The statistical average processing time of this task is 26.97 ms. Finally, we come across that initialization and verification on the Android NFC-enabled smart phone is very fast, and the impact on the authentication protocol with single APDU is very small.

5.4.3 Read Keys and ECC paramters

Montgomery multiplication algorithm 7 requires some precalculated parameters to be stored on the security controller. In the case of any request sent from the NFC-enabled smart phone to read or modify these parameters, Infineon security controller provides a special APDU for this purpose. This command is only implemented for the test case for this master's thesis. Due to the security aspects in the future this APDU must be disabled.

Figure 5.14 shows the timing analysis required by the Android NFC-enabled smart phone to read the parameters mentioned above, where the minimum time is approximately 19.92 ms and the maximum is 40.35 ms. The average time to read the data on the security controller is 24.34 ms.

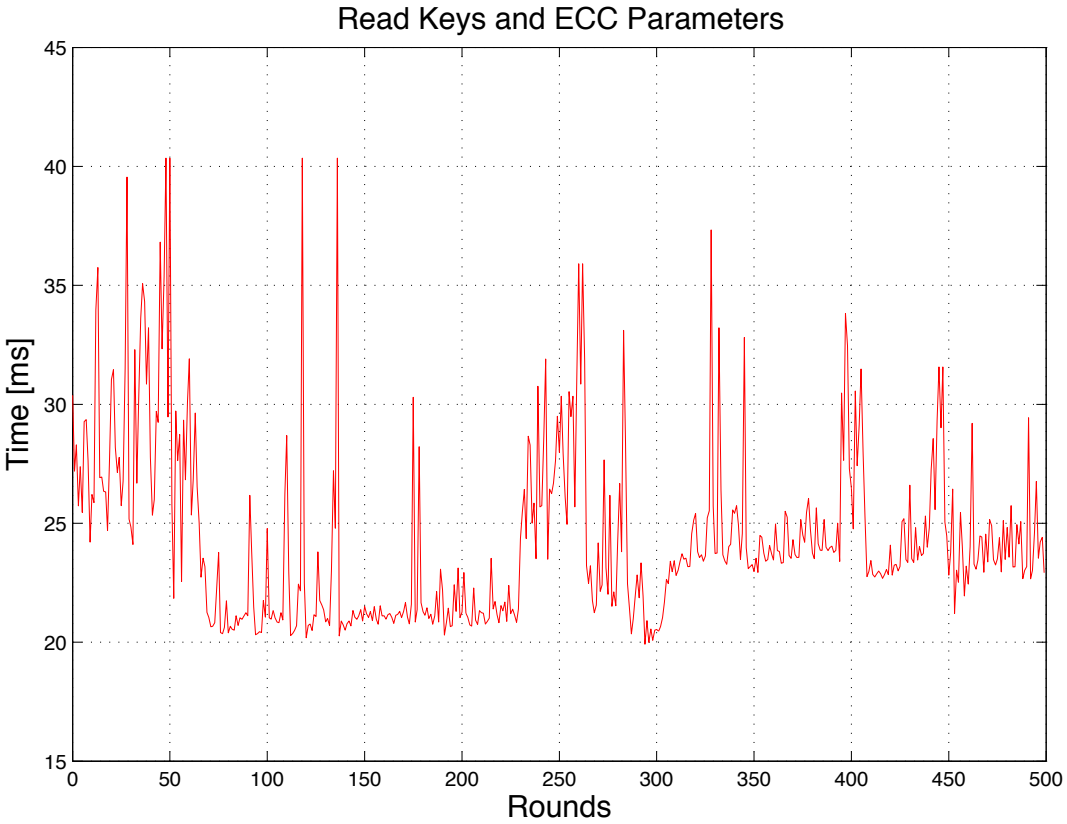


Figure 5.14: Timing Analysis: Read Keys and ECC parameters

Chapter 6

Conclusion

NFC-enabled smart phones provide a very efficient capability to process different cryptosystems in any software even that energy consumption remains still as the main drawback for those smart phones. Integration of the NFC interface into smart phones brought a new evaluation to the embedded devices, whereby customers use NFC-enabled smart phones for different purposes e.g. wallet with various cards.

Many of the efforts are still facing with a very famous negative approach, the *RFID tag cloning*. This issue remains as the next challenge for the cryptographer in the future. In this master's thesis, an efficient way to avoid the cloning of the security controller is provided by the implementation of an optimized ECC-based authentication protocol, which is a challenge-response procedure based on an elliptic curve Diffie-Hellman key exchange. As an algorithm integrated within this protocol, is a Montgomery Multiplication Algorithm defined over prime field. The reason of integrating this algorithm is that there is only a single point multiplication on the security controller and this security controller supports also SLE, which makes ECC calculations defined over prime field more efficient and suitable than ECC calculations defined over binary field.
The initial step of this master's thesis is a high-level software simulation of an authentication protocol.

The software simulation for this protocol is implemented for the elliptic curves defined over prime field and binary field. A comparison of a timing analysis for software simulation of the Montgomery multiplication algorithm of elliptic curves defined over prime field and binary field is applied. Where elliptic curve defined over prime field with 192-bit key length shows a better performance and a faster processing time in comparison to the other elliptic curves defined over prime field with a key length of 224- or 512-bit), and also to the 163-bit elliptic curve defined over binary field. The minimum simulation time of an authentication protocol for 192-bit elliptic curve defined over prime field is 0.69 ms. Furthermore, a timing analysis is elaborated for the field arithmetic operations with different length of operands, for the three different elliptic curves defined over prime field, whereby again 192-bit elliptic curve shows a very efficient processing time. The software simulation of an authentication protocol for the elliptic curve defined over binary field is evaluated too. An authentication protocol for the elliptic curve defined over binary field requires an approximately minimum time of 21 s, which is extremely slower than the elliptic curves defined over prime field.

The is due to the elliptic point multiplication, which takes nearly 4 times longer than a Montgomery multiplication algorithm defined over binary field.

After that, a first communication between an Android NFC-enabled smart phone and the security controller is established, whereby various data are exchanged. A Montgomery multiplication algorithm is computed on the security controller with the help of multiple APDUs.

The computation of a Montgomery multiplication algorithm with multiple APDUs is achieved in an approximate minimum time of 150.85 ms.
An optimized ECC-based authentication protocol between an Android NFC-enabled smart phone and the security controller is also realized with the help of multiple APDUs, which differs hardly from the computation of the Montgomery multiplication algorithm with multiple APDUs. For a 192-bit elliptic curve, an authentication protocol with multiple APDUs is achieved in a minimum time of 156.34 ms, which means the reader initialization and the verification of an authenticity on the reader side is very fast and has a low impact on the processing time.
Furthermore, a Montgomery multiplication algorithm is computed on the security controller with the help of a single APDU, and the minimum computation time for the algorithm is achieved approximately within 21.26 ms.
Finally, the aim of this master thesis is achieved for an authentication protocol with a single APDU. The challenge is computed on an Android NFC-enabled smart phone and is further sent to the security controller, for a single point multiplication. After that, the security controller responses to the Android NFC-enabled smart phone with its public key and also affine x-coordinates, which is computed by the Montgomery multiplication algorithm. The reader initialization and verification appears as mentioned above on the Android NFC-enabled smart phone. An optimized ECC-based Authentication protocol with a single APDU is achieved in approximated minimum time of 22.03 ms.

Chapter 7

Future Work

The ECC-based authentication protocol implemented in this master thesis, can be further extended in the following points:

7.1 Power Analysis

An evaluation of the timing analysis for the optimized ECC-based authentication protocol and performance behavior of reader and security controller is realized in this master's thesis. Therefore, the evaluation of the required energy for this authentication protocol might be an interesting option for the future work.

7.2 Android Back-End Server Communication

For a purpose of an authentication protocol only a communication between an Android NFC-enabled smart phone and the security controller is established. Therefore, no secured communication channel between a reader and server is provided. The secured channel communication between reader and server might be used to manage the properties of the tag e.g. check an authenticity of the product.

7.3 Other NFC-enabled Smart Phones

The authentication protocol for the security controller is only implemented for the Android based OS embedded devices. However, this optimized ECC-based authentication protocol can be further implemented for other NFC-enabled embedded devices such as Windows phone, BlackBerry etc.

7.4 Timing Attacks

The Montgomery multiplication algorithm defined over prime field is widely used for an optimized authentication protocol based on DH-key exchange. It is considered as one of the most efficient algorithm for computing modular multiplications without using one of the most expensive operation in cryptography, *division*. The **Final Subtraction** of

Montgomery Multiplication Algorithm considering the `if` statement, determines a very important decision for the security aspect. Whereby this final subtraction depends on the message and the secret bit [SST04]. Observing the distribution of the final subtraction can help an attacker to guess the secret bit.

7.5 Privacy Enhanced Tag Authentication

The public-key signature generation algorithm and the verification algorithm for the signature is left out of the protocol in this master's thesis. Although, this is necessary for the following step.
The authentication protocol of the security controller described in this master's thesis can be further extended to a privacy enhanced protocol [BHM08]. For instance, the protection of the location and the forward location privacy. Further information about implementation of the privacy enhanced tag authentication is available at [BHM08].

Bibliography

[1.] SEC 1. Elliptic Curve Cryptography. Standards for Efficient Cryptography Group. `http://www.secg.org`, October 2013.

[14401] ISO. ISO/IEC 14443-3:2011. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 3: Initialization and anticollision, 2001.

[14408a] ISO. ISO/IEC 14443-1:2008. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 1: Physical characteristics, 2008.

[14408b] ISO. ISO/IEC 14443-4:2008. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 4: Transmission protocol, 2008.

[14410] ISO. ISO/IEC 14443-2:2010. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 2: Radio frequency power and signal interface, 2010.

[78113] ISO. ISO/IEC 7816-4:2013. Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange, 2013.

[AAE+13] Ali Alzahrani, Abdullah Alqhtani, Haytham Elmiligi, Fayez Gebali, and Mohamed S. Yasein. NFC Security Analysis and Vulnerabilities in Healthcare Applications. In *IEEEPacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 302–305, Aug 2013.

[aes01] Announcing the Advanced Encryption Standard (AES). Federal Information Proccesing Standards Publication 197. United States National Institute of Standards and Technology (NIST), November 26, 2001.

[AG] Infineon Technologies AG. SLE 77 SOLID FLASH^{TM} $family$: 16 − $bit\ security\ controller, March$, 2014.

[AGC10] Mete Akgün, A.Özgan Gürel, and M.Ufuk Caglayan. Attacks to a lightweight RFID mutual authentication protocol. In *International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 1–5, Nov 2010.

[AGGB11] B.R. Ambedkar, Ashwani Grupta, Praktiksha Gautam, and S. S. Bedi. An Efficient Method to Factorize the RSA Public Key Encryption.

Page(s): 108 - 111 E-ISBN : 978-0-7695-4437-3 Print ISBN: 978-1-4577-0543-4, Katra, Jammu, 2011. Internationl Conference on Communication Systems and Network Technologies.

[aI77] Developed at IBM. Data Encryption Standard. In *In FIPS PUB 46, Federal Information Processing Standards Publication*, pages 46–2, 1977.

[alo] planetmath.org alozano. Weierstrass equation of an elliptic curve. `http://planetmath.org/weierstrassequationofanellipticcurve`, November, 2013.

[ARH08] Sheikh Iqbal Ahamed, Farzana Rahman, and Md. Endadul Hoque. ERAP: ECC based RFID Authentication Protocol. pages 219–225. 12th IEEE International Workshop on Future Trends of Distributed Computing Systems, 21-23 Oct. 2008.

[ASa] National Security Agency and Central Security Servce. The Case for Elliptic Curve Cryptography. `http://www.nsa.gov/business/programs/elliptic_curve.shtml`, November, 2013.

[ASb] National Security Agency and Central Security Servce. Cryptoanalysis/Signals Analysis. `http://www.nsa.gov/careers/career_fields/cryptsiganalysis.shtml`, November, 2013.

[Asc13] Ascendia. Near Field Communications - Comparison with other technologies. http://www.ascendia.ro/work/eLearning/mobile_nfc/#11, December 2013.

[Bar86] Paul Barrett. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. pages 311–323. Advances in Cryptology - CRYPTO'86. Lecture Notes in Computer Science, 1986.

[Bas13] BSc Rejhan Bashagic. Design and Implementation of an NFC Interface for Home Appliances and Consumer Electronics. Master's thesis, Institute for Technical Informatics, Graz University of Technology, May, 2013.

[BBD+] Holger Bock, Michael Braun, Markus Dichtl, Erwin Hess, Johann Heyszl, Walter Kargl, Helmut Koroschetz, Bernd Meyer, and Hermann Seuschek. A Milestone Towards RFID Products Offering Asymmetric Authentication Based on Elliptic Curve Cryptography. Invited talk at RFIDsec 08.

[BDD] Mustapha Benssalah, Mustapha Djeddou, and Karim Drouiche. Efficient ECC implementation architecture suitable for RFID technology. pages 1–4. 24 International Conference on Microelectronics (ISM), 2012.

[BHLM01] Michael Brown, Darrel Hankerson, Julio Lopez, and Alfred Menzes. Software Implementation of the NIST Elliptic Curves Over Prime Fields. pages 250–265. Springer Berlin Heidelberg, 8-12 April 2001. Lecture Notes in Computer Science.

[BHM08] Michael Braun, Erwin Hess, and Bernd Meyer. Using Elliptic Curves on RFID Tags. *International Journal of Computer Science and Network Security (IJCSNS) 02/2008*, 8:1–9, February 2008.

[Ble] Bletechlypark. MACHINES BEHIND THE CODES. `http://www.bletchleypark.org.uk/content/machines.rhtm`, November, 2013.

[BTW+13] Christoph Busold, Ahmed Taha, Christian Wachsmann, Alexandra Dmitrienko, Hervé Seudié, Majid Sobhani, and Ahmad-Reza Sadeghi. Smart Keys for Cyber-cars: Secure Smartphone-based NFC-enabled Car Immobilizer. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, CODASPY '13, pages 233–242, San Antonio, Texas, USA, 2013. New York, NY, USA, ACM.

[Cas] Wouter Castryck. Forms of elliptic curves. `https://www.cosic.esat.kuleuven.be/bcrypt/lecture%20slides/wouter.pdf`, November, 2013.

[CK13] Byungrae Cha and Jongwon Kim. Design of NFC Based Micro-payment to Support MD Authentication and Privacy for Trade Safety on NFC Applicaitons. In *Seventh International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pages 710–713, 3-5 July 2013.

[Cla96] John Clark. Attacking authentication protocols. *High Integrity Systems*, 1, 1996.

[CMO98] Henri Cohen, Atsuko Miyaji, and Takatoshi Ono. Efficient Elliptic Curve Exponentation Using Mixed Coordinates. In *Advances in Cryptography - ASIACRYPTO '98*, pages 51–65. Springer-Berlin Heidelber, 1998.

[Cor05] ATMEL Corporation. Understanding the Requirements of ISO/IEC14443 for Type B Proximity Contactless Identification Cards. Technical report, 2005.

[Dev13] Android Developers. Android, the world's most popular mobile platform. `http://developer.android.com/about/index.html`, Last Checked: November, 2013.

[DH76] Whitefield Diffie and Martin E. Hellman. New Directions in Cryptography. pages 644–654. IEEE Transactions on Information Theory, 1976.

[DKL+98] Jean-François Dhem, François Koeune, Philippe-Alexandre Leroux, Patrick Mestré, Jean-Jacques Quisquater, and Jean-Louis Willems. A Practical Implementation of the Timing Attack. In Jean-Jacques Quisquater and Bruce Schneier, editors, *CARDIS*, volume 1820 of *Lecture Notes in Computer Science*, pages 167–182. Springer, 1998.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamire. Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In *ASIACRYPT*, Lecture Notes in Computer Science, pages 158–176. Springer, 2010.

[DMB+12] Norbert Druml, Manuel Menghin, Rejhan Basagic, Christian Steger, Reinhold. Weiss, Holger Bock, and Josef Haid. NIZE - a Near Field Communication interface enabling zero energy standby for everyday electronic devices. In *IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 261–267, 2012.

[Doc] Oracle Java SE Documentation. Java Cryptography Architecture (JCA) Reference Guide. `http://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html`, Last Checked: Februar 2013.

[DR08] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. In *IETF RFC 4346*, August 2008.

[DRR13] Qian Ding, Trey Reece, and William H. Robinson. Timing Analysis in Software and Hardware to Implement NIST Elliptic Curves over Prime Fields. pages 1358 – 1362. IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS), 4-7 Aug. 2013.

[EA10] I. Erguler and E. Anarim. Attacks on an Efficient RFID Authentication Protocol. In *2010 IEEE 10th International Conference on Computer and Information Technology (CIT)*, pages 1065–1069, June 2010.

[edu] edureka. The Beginner's Guid to Android: Android Architecture. `http://www.edureka.in/blog/beginners-guide-android-architecture/`, Last Checked: November, 2013.

[Ele13] Nikolay Elenkov. Using ECDH on Android. `http://nelenkov.blogspot.co.at/2011/12/using-ecdh-on-android.html`, Last Checked: November 2013.

[ElG85] Taher ElGamal. A Public Key Cryptosystem and a Cignature Scheme Based on Discrete Logarithms. volume IT-31, pages 469–472. IEEE TRANSACTIONS ON INFORMATION THEORY, 1985.

[Elg13] Ben Elgin. Google Buys Android for Its Mobile Arsenal. `http://www.webcitation.org/5wk7sIvVb`, Last Checked: November, 2013.

[ELS+13] Hasoo Eun, Hoonjung Lee, Junggab Son, Sangjin Kim, and Heekuck Oh. Conditional Privacy Preserving Security Protocol for NFC Applications. In *IEEE International Conference on Consumer Electronics (ICCE)*, volume 59, pages 153–160, 2013.

[EMRY13] Ahmed Elbagoury, Ahmed Mohsen, Mohamed Ramadan, and Moustafa Youssef. Practical Provably Secure Key Sharing for Near Field Communication Devices. In *International Conference on Computing, Networking and Communications (ICNC)*, pages 750–755, Jan 2013.

[fECS00] Standards for Efficient Cryptography (SEC 2). Recommended Elliptic Curve Domain Parameters. Version 1.0, Certicom Research, secg-talk@lists.certicom.com, September 20, 2000.

[Fel04] Martin Feldhofer. An authentication protocol in a security layer for RFID smart tags. In *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference - MELECON*, volume 2, pages 759–762. IEEE Computer Society, 2004.

[Fin10] Klaus Finkenzeller. *RFID Handbook, Fundametals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. ISBN: 978-0-470-69506-7. June 2010.

[FKK+08] Rainer Falk, Florian Kohlmayer, Andreas Koepf, Michael Braun, Hermann Seuschek, and Mingyan Li. Application of Passive Asymmetric RFID Tags in a High-Assurance Avionics Multi-Domain RFID Processing System. In *4th European Workshop on RFID Systems and Technologies (RFID SysTech)*, pages 1–7, June 2008.

[For13a] NFC Forum. The NFC Forum. `http://www.nfc-forum.org/about-us/`, Last Checked: November 2013.

[For13b] NFC Forum. NFC Forum Technical Specifications - NFC Forum Tag Type Technical Specifications. `http://www.nfc-forum.org/specs/spec_list/`, Last Checked: November 2013.

[Gao93] Shuhong Gao. *Normal Bases over Finite Fields*. PhD thesis, University of Waterloo, 1993.

[GKPP06] Jorge Guajardo, Sandeep S. Kumar, Christof Paar, and Jan Pelzl. Efficient Software-Implementation of Finite Fields with Applications to Cryptography. *Acta Applicandae Mathematica*, 93(1-3):3–32, 2006.

[HB] Ernst Haselsteiner and Klements Breitfluß. *Security in Near Field Communication Strengths and Weakness*. Philips Semiconductors, Mikronweg 1, 8101 Gratkorn, Austria.

[Hei] Infineon Chip Card & Security Target Lite: Steffen Heinkel. M7794 A12 and G12. `https://www.commoncriteriaportal.org/files/epfiles/0883b_pdf.pdf`, 08–09–2013.

[HFP10] Michael Hutter, Martin Feldhofer, and Thomas Plos. An ECDSA Processor for RFID Authentication. In *Radio Frequency Identification: Security and Privacy Issues*, pages 189–202. Springer-Berlin Heidelber, 2010.

[HHM00] Darrel Hankerson, Julio Lopez Hernandez, and Alfred J. Menezes. Software Implementation of Elliptic Cure Cryptography over Binary Fields. In *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems*, pages 1–24. Springer-Verlag, 2000.

[HJWH11] Yu-Jung Huang, Chi-Hung Jiang, Hsuan-Hsun Wu, and Yi-Hao Hong. Mutual Authentication Protocol for RFID System. pages 73–80. IEEE 14th International Conference on Computation Science and Engineering (CSE), 24-26 Aug. 2011.

[HKhF11] Ondrej Hyncicaa, Pavel Kucera, Petr honzik, and Petr Fiedler. Performance Evaluation of Symmetric Cryptography in Embedded Systems. In *The 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, volume 2, pages 277–282, Prague, Czech Republic, 2011.

[HLL12] Yu-Jung Huang, Wei-Cheng Lin, and Hung-Lin Li. Efficient Implementation of RFID Mutual Authentication Protocol. *IEEE Transactions on Industrial Electronics*, 59(12):4784–4791, Dec 2012.

[HMV04] Darrel Hankerson, Alfred Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. ISBN 0-387-95273-X. Springer-Verlag New York, Inc., 2004.

[Höl13] Andrea Höller. Elliptic Curve Cryptography on a Lightweight Microprocessor for RFID. Master's thesis, Institute for Technical Informatics, Graz University of Technology, March, 2013.

[Hut] Michael Hutter. RFID Authentication Protocols based on Elliptic Curves: A Top-Down Evaluation Survey. pages 101 – 110. International Conference on Security and Cryptography - SECRYPT, it is part of ICETE - Proceedings of the International Joint Conference on e-Business and Telecommunications, Milan, Italy, July 7-10, 2009,.

[HZB+13] Gesine Hinterwälder, Christian T. Zenger, Foteini Baldimtsi, Anna Lysyanskaya, Christof Paar, and Wayne P. Burleson. Efficient E-Cash in Practice: NFC-Based Payments for Public Transportation Systems. In *Privacy Enhancing Technologies*, volume 40-59. 13th International Symposium, PETS, Springer Berlin Heidelberg, July 10-12, 2013.

[Inc06] NFC Forum. Inc. NFC Data Exchange Format (NDEF). Technical Specification. NDEF 1.0 , 2006.

[JF09] Zhang Juan and Deng Fangmin. The Authentication and Key Agreement Protocol Based on ECC for Wireless Communications. pages 1–4. International Conference Management and Service Science, 20-22 Sept. 2009.

[Kob87] Neal Koblitz. Elliptic Curve Cryptosystems. volume 48, pages 203–209. Mathematics of Computation, January 1987.

[Kob92] Neal Koblitz. CM-Curves with Good Cryptographic Properties. In *Advances in Cryptology CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 279–287. Springer Berlin Heidelberg, 1992.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113, London, UK, UK, 1996. Springer-Verlag.

[KP04] Heiko Knospe and Hartmut Pohl. {RFID} security. *Information Security Technical Report*, 9(4):39 – 50, 2004.

[Li10] Nan Li. Research on Diffie-Hellman key exchange. pages V4–634–V4–637. 2nd International Conference On Computer Engineering and Technology (ICCET), 16-18 April 2010.

[Lin] LinuxConfig.org. Android system architecture. `http://linuxconfig.org/android-system-architecture`, Last Checked: November 2013.

[Low95] Gavin Lowe. An Attack on the Needham-Schroeder Public-key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, November 1995.

[LSL+] Yu-Cheng Lin, Yu-Chihu Su, Nan-Hai Lo, Weng Fang Cheung, and Yen-Pei Chen. Application of Mobile RFID-Based Safety Inspection Management at Construction Jobsite. http://www.intechopen.com/download/get/type/pdfs/id/45004, November, 2013.

[LTD] TOPPAN FORMS CO LTD. The 3 Modes of NFC. `http://www.nfc-world.com/en/about/02.html`, October 2013.

[LWaYX08] Peng Luo, Xinan Wang, and Jun GEnd an Ying Xu. Low-power hardware implementation of ECC processors suitable for low-cost RFID tags. In *9th International Conference on Solid-State and Integrated-Circuit Technology (ICSICT)*, pages 1681–1684, 2008.

[Mat] MathWorks. Matlab-The Langauge of Technical Computing. www.matlab.com, February, 2014.

[MDS+13] Manuel Menghin, Norbert Druml, Christian Steger, Reinhold Weiss, Holger Bock, and Josef Haid. Using field strength scaling to save energy in mobile hf-band rfid-systems. *EURASIP Journal on Embedded Systems*, 2013:4, 2013.

[MF06] W. Simpson M. Friedl, N. Provos. Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol. Network Working Group, March 2006.

[MIF] MIFARE. MIFARE SmartCard IC's. `http://www.mifare.net/en/products/mifare-smartcard-ic-s/`.

[MIF13] MIFARE. The success of MIFARE. `http://www.mifare.net/en/home/`, November 2013.

[Mil86] Victor S. Miller. Use of Elliptic Curves in Cryptography. In *Advances in Cryptography - CRYPTO '85 Proceedings*, volume 218 of *0302-9743*, pages 417–426. Spinrger Berlin Heiderlberg, 1986.

[Mol] David Molnar. The RFID Hacking Underground. http://www.wired.com/wired/archive/14.05/rfid.html, February, 2014.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. volume 44, pages 519–521. Mathematics of Computation, 1985.

[MR] Rob van der Meulen and Janess Rivera. Gartner Says Smartphone Sales Accounted for 55 Percent of Overall Mobile Phone Sales in Third Quarter of 2013. http://www.gartner.com/newsroom/id/2623415, February, 2014.

[MR09] Marc Pasquet Marie Reveilhac. Promising Secure Element Alternatives for NFC Technology. pages 75–80. First International Workshop on Near Field Communication, 2009.

[MS13] Anoop MS. Elliptic Curve Cryptography. An Implementation Tutorial. Available at: `http://www.infosecwriters.com/text_resources/pdf/Elliptic_Curve_AnnopMS.pdf`, November, 2013.

[Mul] Coling Mulliner. Vulnerability Analysis and Attacks on NFC-enabled Mobile Phones. In *2009 International Conference on Availability, Reliability and Security.*

[Mur] David Murphy. 1 in 5 Smartphones NFC-enabled by 2014. http://mobilemarketingmagazine.com/1-5-smartphones-nfc-enabled-2014/, Feburary, 2014.

[MvOV01] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* ISBN: 0-8493-8523-7. CRC Press, Fifth Printing (August 2001).

[MW00] Ueli M. Maurer and Stefan Wolf. The Diffie-Hellman Protocol. volume 19, pages 147–171. Kluwer Academic Publishers, 2000.

[Nea13] NearFieldCommunication.org. Security Risks of Near Field Communication. `http://www.nearfieldcommunication.org/nfc-security-risks.html`, November, 2013 November, 2013.

[NFC] NFCNearFieldCommunications.org. Near field Communication Security Risks. http://www.nfcnearfieldcommunication.org/nfc-security-risks.html, Last Checked.

[OHA13a] Open Handset Alliance. Android,. `http://www.openhandsetalliance.com/android_overview.html`, Last Checked: November, 2013.

[OHA13b] Open Handset Alliance. Open Handset Alliance. `http://www.openhandsetalliance.com/oha_faq.html`, Last Checked: November, 2013.

[Oka88] Tatsuaki Okamoto. *Encrytion and authentication schemes based on public-key systems.* PhD thesis, The Universtiy of Tokyo, 1988.

[Oraa] Oracle. java.math Class BigInteger. `http://docs.oracle.com/javase/6/docs/api/java/math/BigInteger.html`, Last Checked: January 2014.

[Orab] Oracle. NetBeans IDE 7.4-Download. https://netbeans.org/downloads/, November, 2013.

[oST92] National Institute of Standards and Technology. The Digital Signature Standard, proposal and discussion, Communications of the ACM. volume 35(7), pages 36–54, 1992.

[P1300] IEEE P1363. Standard Specifications for Public-Key Cryptography. Technical report, Institute of Electrical and Electronics Engineers, 2000.

[PBJ06] CAS/IDE PHILIPS. Bob Jiang. *NFC vs ISO14443 vs Felica.* Feb 23, 2006.

[PCTRFSE13] Ashwin Pal (CSO The Resource For Security Executives). Near Field Communication - the scurity risks. `http://www.cso.com.au/article/440741/near_field_communication_security_risks_/`, November, 2013 November, 2013.

[PHF+13] Thomas Plos, Michael Hutter, Martin Feldhofer, Maksimiljan Stiglic, and Francesco Cavaliere. Security-Enabled Near-Field Communication Tag With Flexible Architecture Supporting Asymmetric Cryptography. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(11):1965–1974, Nov 2013.

[PO08] Pantelimon George Popescu and Sanda Osiceanu. The discrete logarithm problem in cyclic subgroups of not necessary cyclic groups. In *1st International Conference on Information Technology, IT 2008.*, pages 1–4, 18-21 May 2008.

[PP10] Christof Paar and Jan Pezl. *Understanding Cryptography.* ISBN 978-3-642-04100-6. Springer Heidelberg Dordrecht London New York, 2010.

[PWmMlZsL11] Shang Ping Wang, Qiao mei Ma, Ya ling Zhang, and You sheng Li. An Authentication Protocol for RFID Tag and Its Simulation. *JNW*, 6(3):446–453, 2011.

[Rad02] Cristian Radu. *Implementing Electronic Card Payment Systems.* ISBN: 1-58053-305-1. Artech House Computer Security Series, 2002.

[RCMSDP13] Antonio Cortina Reyes, Ana Karin Vega Castillo, Miguel Morales-Sandova, and Arturo Diaz-Perez. A Performance Comparison of Elliptic Curve Scalar Multiplication Algorithms on Smartphones. In *International Conference on Electronics, Communications and Computing (CONIELECOMP)*, pages 114–119, March 2013.

[Riv] Ronald L. Rivest. "Cryptology". In J.Van Leeuwen Handbook of Theoretical Computer Science1. 1990.

[RLS13] Michael Roland, Josef Langer, and Josef Scharinger. Applying Relay Attacks to Google Wallet. In *5th International Workshop on Near Field Communication (NFC)*, pages 1–6, Feb 2013.

[Rob] Mark Roberti. The History of RFID Technology. http://www.rfidjournal.com/articles/view?1338, Last Checked: December 2013.

[Ros] Michael Rosing. Implementing Elliptic Curve Cryptography. http://www.manning.com/rosing/, December, 2013.

[Ros99] Michael Rosing. *Implementing Elliptic Curve Cryptography*. Manning Publications Co., Greenwich, CT, USA, 1999.

[RS] Felix Richter and Statista.com. Android Accounts for 70% of Smartphone Sales in China. http://econintersect.com/b2evolution/blog1.php/2013/09/06/android-accounts-for-70-of-smartphone-sales-in-china, March, 2014.

[RSS] Margaret Rouse (Search Security). RSA algorithm (Rivest-Shamir-Adleman). http://searchsecurity.techtarget.com/definition/RSA, November, 2013.

[Sho97] Victor Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Advances in Cryptography - EUROCRYPT' 97*, pages 256–266. Springer Berlin Heidelberg, 1997.

[SMD] Ass.Prof. Dip.-Ing. Dr.techn. Christian Steger, Dip.-Ing. BSc Manuel Menghin, and Dip.-Ing. BSc Norbert Druml. META[:SEC:]- Mobile Energy-efficient Trustworthy Authentication Systems with Elliptic Curve based SECurity. `http://www.iti.tugraz.at/cms/index.php?option=com_jresearch&view=project&task=show&id=17`, 2013.

[SST04] Hisayoshi Sato, Daniel Schepers, and Tsuyoshi Takagi. Exact analysis of montgomery multiplication. In *Proceedings of the 5th International Conference on Cryptology in India*, INDOCRYPT'04, pages 290–304, Berlin, Heidelberg, 2004. Springer-Verlag.

[SY08] Zhijie Jerry Shi and Hai Yan. Software Implementation of Elliptic Curve Cryptography. volume 7, pages 141–150. International Journal of Network Security, July 2008.

[TF13] BSc Trebo Fioriello, Manuel. Trusted Device Interaction over NFC. Master's thesis, Technical University of Graz, July, 2013.

[TFHA+11] Jonathan Taverne, Armando Faz-Hernández, Diego F. Aranha, Francisco Rodríguez-Henríquez, Darrel Hankerson, and Julio López. Software implementation of binary elliptic curves: impact of the carry-less multiplier on scalar multiplication. *IACR Cryptology ePrint Archive*, 2011:170, 2011.

[Tut13] Simply Easy Learning Tutorialspoint. Android Architecture. `http://www.tutorialspoint.com/android/android_architecture.htm`, November, 2013 November, 2013.

[Tyl] Robert Tyley. Spongy Castle repackage of Boncy Castle for Android. http://rtyley.github.io/spongycastle/, November 2013.

[Vad13] Prof. Salih Vadhan. AM 106/206 Applied Algebra: Lecture Notes 7. `http://people.seas.harvard.edu/~%20salil/am106/fall10/Cyclic.pdf`, November, 2013.

[Vio] Bob Violino. What is RFID. `http://www.rfidjournal.com/articles/view?1339`, Last checked: December 2013. RFID Journal.

[VK11] Roel. Verdult and François. Kooman. Practical Attacks on NFC Enabled Cell Phones. In *3rd International Workshop on Near Field Communication (NFC)*, pages 77–82, Feb 2011.

[War13] Christina Warre. Google Reveals Mobile Payment System: Google Wallet. `http://mashable.com/2011/05/26/google-mobile-payment-system-liveblog/`, November 2013.

[X9.13a] ANSI X9.62-1998:. Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA). Technical report, American Bankers Association, 2013.

[X9.13b] ANSI X9.63-1998:. Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Key Agreement and Key Transport Schemes. Technical report, American Bankers Association, December, 2013.

[Zc09] Jun Zhang and LiQun chen. An Improved Algorithm For Discrete Logarithm Problem. In *International Conference on Environmental Science and Information Application Technology (ESIAT)*, volume 2, pages 658–661, 2009.

[ZZ09] Liangbin Zheng and Yongbin Zhang. An Enhanced IPSec Security Strategy. In *International Forum on Information Technology and Applications, IFITA '09.*, volume 2, pages 499–502, May 2009.