FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

On Austrian ePassport Security

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Stefan Vogl, BSc
Matrikelnummer 0926857

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao. Univ. Prof. Dr. Wolfgang Kastner
Mitwirkung: Dr. Christian Platzer

Wien, 07.08.2014

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

On Austrian ePassport Security

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Stefan Vogl, BSc
Registration Number 0926857

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao. Univ. Prof. Dr. Wolfgang Kastner
Assistance: Dr. Christian Platzer

Vienna, 07.08.2014

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Stefan Vogl, BSc
Inzersdorfer Straße 80/1/13, 1100 Wien

Abstract

In 2006, the first passports with an embedded RFID (Radio Frequency Identification) chip were released in Austria. Since then, all data printed on the passport, except the holder's size and signature, are additionally stored on this chip. This includes the name, date of birth and the picture printed on the data page. Since 2009, two fingerprints are stored on the chip as well. Due to the fact that the mentioned data are very sensitive, it should be well protected. In a worst-case scenario, an attacker would use his NFC-enabled (Near Field Communication) smartphone to read, replay or modify these data for malicious purpose. Currently, various mechanisms are implemented to secure this information. These mechanisms try to prevent attacks against RFID systems like skimming, cloning and data modification.

This thesis provides an in-depth analysis of the security measurements implemented in the Austrian ePassport. After this analysis, a feasibility study is done on the found weaknesses. This study shows, which of the weaknesses can be practically exploited and what threats can arise of this exploitation. For practically exploitable weaknesses, a proof of concept is developed, whereas their emphasis lies on the availability of the hardware to show that no specialized equipment is necessary to attack ePassports.

The results of this thesis show, that attacks are possible with consumer grade hardware, especially if taking into account special circumstances in the Austrian implementation of the ePassport. Some attacks can even be implemented on an NFC-enabled off-the-shelf smartphone.

Kurzfassung

In Österreich wurden 2006 die ersten Reisepässe, ausgestattet mit einem RFID (Radio Frequency Identification) Chip, eingeführt. Seitdem sind die Daten, wie zum Beispiel Name, Geburtsdatum und das auf dem Reisepass aufgedruckte Foto zusätzlich auf dem Chip abgelegt. Seit 2009 werden außerdem noch zwei Fingerabdrücke auf dem Chip gespeichert. Da es sich hierbei um äußerst sensible Daten handelt, müssen diese besonders geschützt werden. Im schlimmsten Fall könnte ein Angreifer mit einem NFC-fähigen (Near Field Communication) Smartphone diese sensiblen Daten unbefugt auslesen, kopieren oder verändern. Um dies zu verhindern, wurden Sicherheitsprotokolle entwickelt und implementiert, die den Reisepass-Chip vor Angriffen, wie Skimming (unbefugtes Auslesen), Klonen und vor dem Verändern der gespeicherten Daten, schützen sollen.

Diese Arbeit bietet eine detaillierte Analyse der im österreichischen Reisepass implementierten Sicherheitsmaßnahmen. Auf Basis dieser Analyse und der gefundenen Schwachstellen wird eine Machbarkeitsstudie durchgeführt, die aufzeigt, welche dieser Schwachstellen in der Praxis ausgenutzt werden können und welche Gefahren daraus entstehen. Die Praktikabilität dieser Angriffe wird durch die Entwicklung eines Proof of Concepts gezeigt. Besonderes Augenmerk bei dieser Entwicklung liegt auf der verwendeten Hardware, um zu zeigen, dass keine speziellen Ressourcen benötigt werden, um den elektronischen Reisepass anzugreifen.

Das Ergebnis dieser Arbeit zeigt, dass Angriffe auf den Reisepass bereits mit handelsüblicher Hardware möglich sind. Größeren Erfolg haben diese, wenn besondere Eigenschaften des österreichischen Reisepasses in Betracht gezogen werden. Einige dieser Angriffe können sogar mit gängigen NFC-fähigen Smartphones durchgeführt werden.

Contents

CHAPTER 1

Introduction

The International Civil Aviation Organization (ICAO), as the aviation agency of the United Nations (UN), has a long history of specifying international travel documents. In the early 1980s, they published the first edition of the ICAO 9303 document specifying ‘A Passport with Machine Readable Capabilities‘. These first machine readable passports were equipped with a Machine Readable Zone (MRZ). In this zone on the data page of the passport the most important data of the passport like the name or the date of birth are printed in an OCR-B font. In 1995, the ICAO recognized the growing importance of biometrics to link the passport tighter to its owner. To store biometric features in a machine readable way on the passport, different storage technologies were examined. In the end, a contactless integrated circuit was chosen for data storage. In the following years, the first specifications for electronic passports were published by the ICAO [1].

After the 9/11 terrorist attacks on the World Trade Center, multiple countries started to implement electronic passports [2]. Austria started to deploy the first generation of electronic passports on June 16, 2006. This first generation included the document holder’s photo as a biometric feature. In 2009, the second generation, which also included an electronic representation of two of the holder’s fingerprints, was deployed.

To ensure the confidentiality, integrity and authenticity of the data stored on the chip the ICAO specifies a set of security mechanisms. Some of these mechanisms are mandatory and some of them are optional. To ensure the security of the additional biometrics like fingerprints, the German *Bundesamt für Sicherheit in der Informationstechnik* (BSI) proposed some extended security features, which were later adopted by the ICAO. In 2004 the European Council passed the Council Regulation No 2252/2004 [3], which made the storage of two fingerprints on the passport chip and the extended security features mandatory.

1.1 Related Work

In the last decade, not only passports have been equipped with chips, to hold the personal data, also printed on the document, but also other high security documents are being equipped with electronic storage. The European Council Directive 2006/126/EC [3] introduced the possibility to integrate a microchip into a European driving license in 2006 and in 2012, Commission Regulation No 383/2012 [4] laid down the technical requirements for an electronic driving license. The microchip should include, besides the data printed on the document, at least the license holder's face image and signature. Two fingerprints and biometric data regarding iris can be stored optionally. It is also allowed, that member states store additional country specific data on the chip. The security features to protect the sensitive data on the chip are identical to the ones used in electronic passports. Basic Access Control (see Section 3.1) is used to protect the general access to the data on the chip, whereas Passive Authentication (see Section 3.2) is used to protect the integrity of the stored data. If more sensitive data is stored on the chip, it has to be protected by Extended Access Control (see Section 3.4). The technical details for electronic driving licenses are defined in ISO 18013. Electronic driving licenses are used for example in Australia and in Mexico. Besides driving licenses, other high security documents, like residents permits and vehicle registration cards are equipped with microchips for electronic storage and protection of the data and the documents. Austria already implemented the electronic resident permits and the electronic vehicle registration card. The electronic residents permit is, like the electronic passport, specified in the [5]. The electronic vehicle registration card has to be, unlike the other described electronic documents, according to European Commission Directive 2003/127/EC [6] a contact-based smartcard.

While microchips are included into official high security documents, to provide a better verifiability and higher security, they are also integrated into credit cards, to provide a better user experience. In 2013, Austrian banks started to issue payment cards (Bankomatkarte) and credit cards with so called Near Field Communication (NFC) chips [7]. NFC enables customers to use their payment cards contactless and up to a specified amount, for example €25, without entering their Personal Identification Number (PIN). The security of these payment cards has already been topic of some research papers. In [8] a downgrading attack is proposed, which allows the cloning of a card. This downgrading attack is only possible if the less secure MagStripe protocol is supported by the card. According to [8] this is not the case for the Austrian contactless payment card (NFC Bankomatkarte).

From the beginning of the issuance of electronic passport there have been major security concerns by information security and privacy experts about the security of the different implementations and the threats they pose to the privacy of the passport holders. In [9], the authors conducted a first security analysis with an emphasize on identifying threats and possible attack vectors of the ePassport specifications in general, US implementation and the Malaysian ePassports, which were not compliant to the ICAO specification at that time. In [10], a security analysis to identify risks and threats to the ePassport system proposed by the ICAO is performed as well. They also suggest some countermeasures against the found attacks, including a rights management system, which would enable countries to give some entities only limited access and others full access to their passports. At the 2007 DefCon Conference, Lukas Grünwald [11] presented similar design flaws in the ICAO specification and proposed an attack on inspection systems using a special photo prepared to exploit a vulnerability in a well known JPEG2000 library. In [12], the first practical implementation of an attack on the Basic Access Control was performed with an offline attack using the COPACOBANA[1] FPGA cluster, which was especially designed for breaking symmetric encryption. In 2008, the Dutch security researcher Jeroen van Beek presented the index manipulation attack described in more detail in Section 5.5 at the Black Hat Conference [13], which makes it possible for an attacker to circumvent anti cloning protections such as Active Authentication. In [14] and in [15], it is shown that it is possible to identify passports from different countries even with randomized UIDs because of error codes they return to specific commands.

Similar to the approach in [12] there have been other projects using FPGAs for attacking cryptographic protocols or calculating cryptographic hashes. In [16] for example an FPGA is used to create rainbow tables to attack encrypted mobile communication. They achieved a speedup of around 9x compared to previous implementations in computing rainbow tables for the A5/3 block cipher used in the Global System for Mobile Communications (GSM). Besides breaking cryptographic protocols, FPGAs are also used to calculate hashes for mining so called crypto currencies like Bitcoins. In [17] it is shown, that FPGAs can speedup the process of mining bitcoins by 500.000x compared to a standard CPU.

Besides FPGAs also GPUs are used for mining Bitcoins. In [17] it is argued, that, even if FPGAs can produce a higher throughput while mining Bitcoins, the Total Cost of Ownership (TCO), which includes the costs of acquisition, as well as operating costs, like power consumption, is lower with GPUs than with FPGAs. General Purpose GPU (GPGPU) computing has become very popular with the advances of brute-force attacks against the encryption of wireless network secured with WPA and WPA2 using preshared keys. One of these brute-forcers was developed by Elcomsoft and is called Elcomsoft Wireless Security Auditor (EWSA)[2]. Elcomsoft claims to achieve a speedup of 320x compared to an Intel Core2 Q9400 Quad-Core CPU.

1.2 Motivation

The first passports in Europe were issued around the early 15th century. In this time safe conduct documents were given to people, to prove their identity and that they are protected by their king, when traveling to foreign lands. With more and more people traveling across nation borders, the process of verifying these letters started to take a lot of time. Therefore, passport checks were not very common in the 19th and early 20th century. Only with the beginning of World War I countries started to check passports at their borders, because of security reasons. This led to a common passport design agreed on by the 'League of Nations' in 1920. With the rising importance of passports, security features, which many of them have already been used in

[1]`http://www.copacobana.org/`

[2]`http://www.elcomsoft.com/ewsa.html`

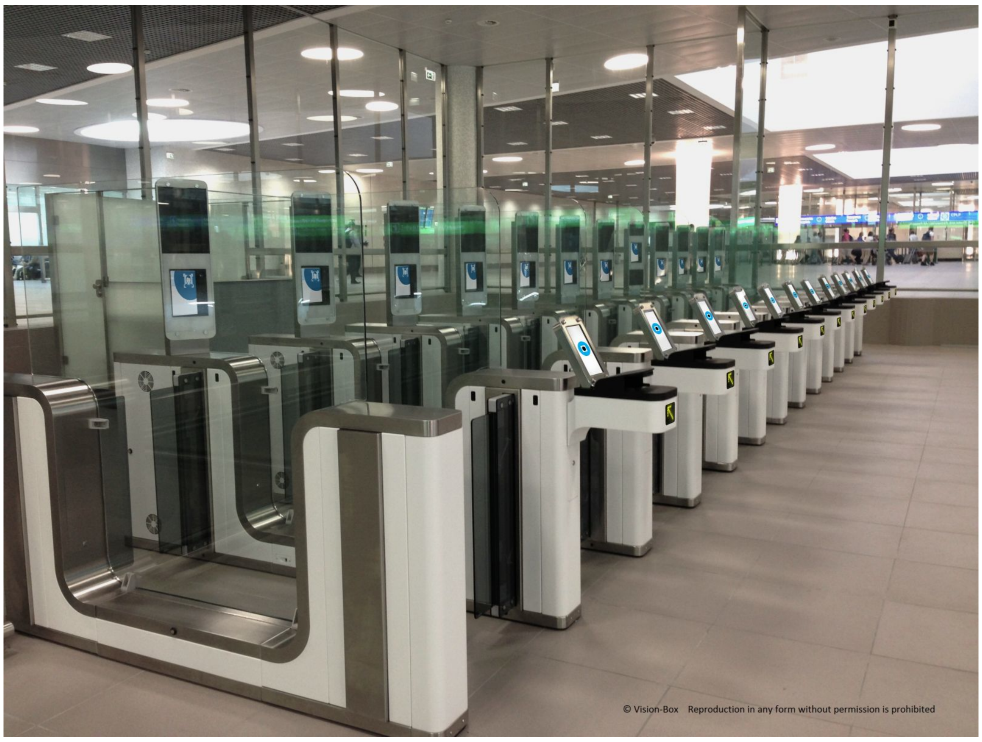

© Vision-Box Reproduction in any form without permission is prohibited

Figure 1.1: eGate at Lisbon International [18]

printing money, were implemented. Besides features for a higher security, means were implemented to guarantee a faster border crossing process. Therefore, the already described MRZ was introduced. For an even higher security, the chip with its additional security features was implemented.

During the first years of electronic passports, only a few countries really checked the chip during the border crossing at their borders, mostly because of the missing hardware at the border. With higher security needs, more borders were equipped with the needed document readers. With the implementation of so called Automated Border Control (ABC) gates (see Figure 1.1) the chip security got more and more important. These ABC gates first read all data from the passport chip and checks the authenticity and integrity via the security features described in this thesis. Based on the outcome of the checks, the passport holder passes on to the next step, where the biometric features stored on the chip are checked, for example through facial recognition, or fingerprint verification software. If in the first step, the checks are already negative, the passport holder is rejected and can't proceed.

If an attacker was able to forge a passport, or alter a genuine passport, so that the stored biometric features are his own, and the gate is not able to detect this alteration, he can cross the border under any assumed identity, as the biometric checks will be positive, because the biometric features stored on the chip are the genuine features of the attacker. Even if an attacker is only able to clone an existing passport without the gate recognizing, he may still be able to cross the border, when he manages to circumvent the biometric checks, for example by a spoofing mask attack as shown in [19].

Although much work in this field has been done already, many proposed attacks are only possible in theory or under very special circumstances. Many papers propose an attack on the Basic Access Control and claim that the entropy of the key is very low, but most attacks need some special knowledge either of the document holder's date of birth or the document number. In [12] for example the attack was proposed only about two years after the deployment of the German ePassport with a very limited amount of issued passports by then. So the document numbers were very limited and the date of expire could only be within two years. On the other side, most of the security analysis done was based either on the Dutch, German, Belgian or the U.S. implementation of the ICAO specification. Little to no research has been done on the Austrian implementation of the electronic passport until now.

1.3 Methodology and Outline

In this thesis, we first do a detailed analysis of the security features specified by the ICAO for the implementation of ePassports. We look at the features, where the specification mandates the used cryptographic algorithms and key lengths and evaluate the security of the chosen parameters. This evaluation especially takes into account the long validity periods of passports. It shows if the chosen algorithms and parameters are considered secure in the future. It is not part of this thesis to find new attacks on the security algorithms, like DES and RSA, themselves.

Based on the conducted research on these international specifications, we analyze the Austrian ePassport implementation. Therefore we collected passports from family, friends and colleagues, with the issuing date evenly distributed over the last eight years. We dumped the content of these passports and looked at the implemented security features, used key lengths and algorithms. Furthermore, we listed the used certificates, to identify when there were changes in used certificates and how long private keys are used compared to the certificate validity period.

Following this analysis we identified attack vectors and threats for ePassport implementations in general. We show how these attack vectors could be exploited and what consequences a successful attack can have on the security of international border crossing procedures. the focus of the evaluation is laid on the impact on the advancing deployments of ABC gates in Europe and all over the world.

In the next step we conducted research on already known attacks. We show how they are feasible in practical environments and how high the probability of a successful attack is. Therefore we took a look at the prerequisites of the attacks and if they are met in the Austrian implementation of ePassports.

Finally we took the most promising attacks and tried to implement proof of concepts for them. For these implementations special emphasis was laid on the used hardware. They show that these attacks are not solely of an academic nature, but can be executed on day-to-day consumer-grade hardware. We implemented a brute-force attack on the Basic Access Control (see Section 3.1) using GPUs, as they are highly optimized for parallel computing and therefore very well suited for the execution of brute-force attacks. As a last attack, we implemented a relay attack, as well as a cloning attack on a low-cost NFC-enabled Android smartphone.

In Chapter 2 we introduce the used technology and some technical background, which is necessary for the understanding of this thesis. Chapter 3 conducts the security analysis of the Austrian ePassport implementation and researches used algorithms and the implemented optional and mandatory security features. In Chapter 4 we describe the attack vectors and the threats they pose. Chapter 5 describes the discovered attacks as well as their implementation and the used technology. Chapter 6 then gives a short outlook of our future work and our conclusion.

CHAPTER 2

Background

This chapter gives an introduction into the technical background necessary to understand this thesis. In the first two sections, we introduce the basic principles and a short history of the RFID technology. In Section 2.4, we give an overview of the RFID technology specified in ISO 14443 and used in electronic passports. In Section 2.5, we introduce the higher layer protocols used by many RFID technologies. Finally in Section 2.6 we describe the filesystem used in electronic passports, the Logical Data Structure (LDS).

2.1 RFID Principles

A Radio Frequency Identification (RFID) system normally consists of RFID tags or transponders, which carry some data, and RFID readers or transceivers, which can read and write data from and to tags. Figure 2.1 shows the basic communication between reader and tag.

RFID tags

RFID tags consist of two main components. An antenna and some integrated circuits for holding and processing data. Tags can either be passive, meaning they have no power source of their own, or active. Active tags have more processing capabilities than passive tags but they are also bigger and more expensive. Passive RFID tags can be built in very small and flexible form factors. These tags can for example be built into textiles or into paper as it is done in electronic passports.

The heart of an RFID tag is its integrated circuit (IC). In passive tags, the IC is powered by the reader via electromagnetic induction. The IC can have many functions. It can either be used as a simple memory which is read and written by the reader, or it can have sophisticated processing capabilities for example for cryptographic calculations. Very often the IC provides a secure storage where for example cryptographic keys can be stored.

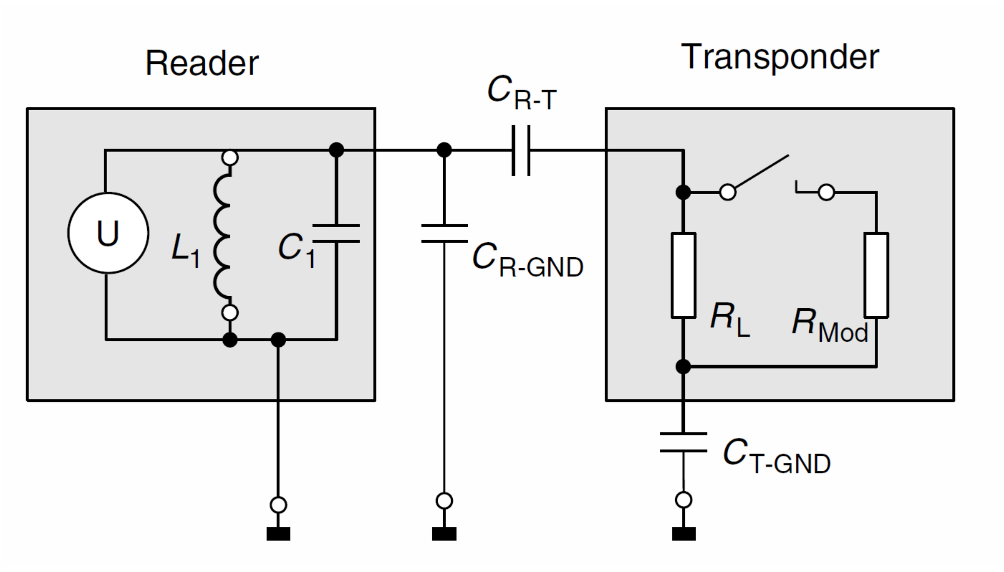


Figure 2.1: Schematics of a basic RFID communication [20]

RFID readers

RFID readers communicate with an RFID tag via a Radio Frequency (RF) channel. With passive tags the reader initiates the channel and sends data. The tag uses the initiated channel to respond to the sent data. If the tag is passive, the energy of the generated magnetic field is used to power the chip.

2.2 Operating Frequencies

RFID systems can operate on different frequencies. Different frequencies are suitable for different purposes. Low frequency (LF) RFID systems operate in the 120 - 140 KHz range. They typically have a reading distance of about 10 - 20 cm and offer a very low data transfer rate because of the low operating frequency. LF systems are very independent of their surrounding environment and can work in the proximity of metal or liquids. Typical purposes for LF systems are access control or car immobilizer systems, where the low reading distance is used as a security feature. LF tags are specified in the ISO 18000-2 standard.

High Frequency (HF) systems operate at a frequency of 13,56 MHz and like LF tags offer a reading distance of about 10 - 20 cm. The higher frequency also offers a higher data transfer rate than in LF systems. The frequency band in which HF systems operate is highly regulated. Typical applications for HF tags are contactless smartcards including electronic passports. HF systems in general are regulated in ISO 18000-3 and contactless smartcards especially in ISO 14443 and ISO 15693.

Ultra High Frequency (UHF) tags are a newer technology. They operate at a frequency of 868 - 928 MHz. The exact frequency used depends on the market. 868 - 870 MHz tags are typically used in Europe whereas in the United States and in Canada tags operating at 902 - 928 MHz are used more often. They offer a reading distance of a few meters depending on their purpose. In the past years, UHF systems are more widely used, mainly because of their low price. UHF tags are defined in ISO 18000-6 and the EPCGlobal standard.

Rather new RFID technologies are Microwave and Ultra Wideband (UWB) systems. Microwave systems operate at a frequency of 2,4 GHz and 5 GHz and offer a reading distance of a few meters up to 30 meters. UWB systems on the other hand do not use a specific frequency, but use multiple frequencies in parallel with a lower signal strength on each frequency. This makes them very useful in sensitive environments like hospitals as they do not interfere with sensitive equipment.

2.3 History of RFID

The first RFID systems are said to have been used during World War II as a way of separating friendly soldiers from enemies. Radar was already used by all parties, but they had no way of separating their own planes from their enemy's. The Germans were the first to find out that the radar signal changes if the pilots roll their plane. They used this fact for differentiating incoming planes. Therefore, they were practically the first to use a passive RFID system. Later, the British put transponders on each plane which started broadcasting a signal when it received a signal from the ground. This was the first active RFID system.

During the 1950s and 1960s, RFID was more and more researched and also used for commercial purposes. One of the first RFID systems were anti theft mechanisms in supermarkets. So called 1-bit systems were used in these cases. A transponder is placed on the merchandise with 1-bit of storage. If a customer pays, the bit is turned off. A reader at the exit detects the tags. If a tag is detected without the bit turned off, the alarm goes off. These systems were very successful and are still used in today's supermarkets. [21]

In the 1970s, the first key card was developed. A reader connected to the door opened this door, if a tag responding with the correct key was detected by the reader, the door would open. These kinds of key cards which are only using an identifier to open a door, are still used for access control in many companies, although they have no security mechanisms and can be cloned very easily.

In the following decades, RFID tags have been used for tagging various things, starting from nuclear materials to cars for automated toll payment systems. These systems all used active transponders. During this time, passive tags were developed for tagging animals. These tags used UHF radio waves to power the tag. The tag then reflected a modulated signal using a technology called backscatter.

During the 1990s, companies started to use the unregulated frequency at 13.56 MHz which offered a better reading range and data transfer than at this time also very popular LF systems provided at 125 kHz. These systems are still popular, especially for contactless smartcards which are used for access control, contactless payment and in electronic passports. Due to their popularity, many of these implementations have been targets of attacks. One of the most prominent successful attack on an HF RFID system was the so called MIFARE hack [22]. The paper shows that there are security flaws in the well-known MIFARE classic contactless smartcard. These flaws can be used to clone cards. MIFARE classic cards are still used for transportation and access control.

In the last decade, UHF systems became more and more popular, because of their reading range and the cheap production prices. The most widespread use for these systems are tags for products using the Electronic Product Code (EPC)[1]. Especially on the American continent UHF systems are very popular for automated toll collection systems. Also the U.S. passport card, which can be used to travel between the U.S., Canada and Mexico makes use of an UHF based RFID chip[2]. [21]

2.4 Proximity Card Standards

Proximity cards, which are often called contactless smartcards, are specified in ISO 14443 [23]. This standard is divided into four parts.

- Part 1: Physical Characteristics
- Part 2: Radio Frequency Power and Signal Interface
- Part 3: Initialization and anticollision
- Part 4: Transmission Protocol

The first part specifies all physical characteristics like the size of the card, the antenna and the placement of the integrated circuit.

The second part defines the power and signal interface. In this part, the parameters of the magnetic field and the used signal modulation is described. The signal interface offers three different bitrates: 106 kbit/s, 212 kbit/s, 424 kbit/s and 848 kbit/s. During the initialization phase 106 kbit/s is used. There are two types of communication signal interfaces defined. Type A uses a 100% Amplitude Shift Keying (ASK) modulation with a Modified Miller encoding for the communication between the reader and the card. For the backward channel, the used modulation and coding depends on the used bitrate. With a bitrate of 106 kbit/s, a load modulation

[1] `http://www.gs1.org/epcglobal`

[2] `http://travel.state.gov/passport/ppt_card/ppt_card_3926.html`

with on / off keying (OOK) with a Manchester coding, is used. Higher frequencies use a load modulation with Binary Phase Shift Keying (BPSK) and Non Return To Zero Level (NRZ-L) coding. Type B uses 10% ASK with Non Return To Zero (NRZ) coding for a forward channel between reader and cards. For the backward channel, load modulation with BPSK and NRZ-L coding is used for all bitrates. For the load modulation, there is always a subcarrier of 848 kHz used. The Austrian ePassport makes use of type A.

Part 3 of the standard describes the initialization and anticollision protocols including the byte format, frames and timing used during initialization and anticollision procedure. The anticollision protocol guarantees the possibility for the reader to communicate with multiple cards in the RF field. The different types of cards concerning the signal interface (Type A and Type B) use different anticollision protocols. To be able to communicate with both types, the reader sends out request commands *REQA* and *REQB* alternating.

- **Anticollision for Type A**
  If a card of type A receives a *REQA* command it answers with an Answer To Request (*ATQA*) including a Unique ID (UID). If multiple cards answer with their *ATQA* the reader can detect a collision. It then sends a *SELECT* command with all the UID bits before the detected collision. Only the card with these UID bits then answers. If there are further collisions, the steps are repeated. If no further collisions are detected by the reader, it sends a select command with the full UID. The UID size can either be single (4 bytes), double (7 bytes) or triple (10 bytes). The UID can be a fixed value or a random number generated by the card as done with the Austrian ePassport. The first byte generally indicates the type of the card. If the first byte is set to 0x08, the following UID is randomly generated.

- **Anticollision for Type B**
  For the anticollision in type B, so called slots are used. The reader sends out a *REQB* with a slot number $N >= 1$. A card receiving the *REQB* calculates a random number between 1 and N. If the calculated number is 1, the card starts to transmit the *ATQB* immediately. If the reader detects a collision it sends a Slot Marker command with the next slot. If it finds a slot with only one card in it, it selects this card. If there is no such slot, it resends the *REQB* command and repeats the procedure until it finds a slot with no collision.

The fourth part specifies the transmission protocols, which apply for type A and B cards. The first part of this protocol is the activation procedure. After the anticollision protocol, the reader can determine if the card is compatible with this protocol. If this is the case it sends a Request Answer To Select (*RATS*) command to the selected card, which then answers with the Answer To Select (*ATS*). These commands carry information about the capabilities of both parties. Using these parameters, the communication channel is set up. The transmission protocol is transparent to the transmitted data. In many cases so-called Application Data Units (APDUs), which are described in more detail in Section 2.5, are transmitted. In general, there are three types of blocks used to transmit data.

- **I-Block**
  Informational block. I-Blocks carry the payload from the application layer. The payload

can be transparently split into multiple I-Blocks. An I-Block can either be a chaining type I-Block, which means there is more data, or a non chaining type, which means there is no more data.

- **R-Block**
  Receive ready block. R-Blocks carry data for error handling.

- **S-Block**
  Supervisory block. S-Blocks are used to exchange control information. One type of S-Blocks is the Waiting Time Extension (WTX), which is described in Section 5.4.

2.5 Application Data Units (APDU)

The general procedure of communicating with a smart card is for a reader to send a command, the card processing it and sending back a response. The messages sent are called Command-APDUs and Response-APDUs. These APDUs are defined in ISO 7816-4 [24]. The general structure of a Command-APDU is shown in Table 2.1.

CLA	INS	P1	P2	L_c	Data	L_e
0	1	2	3	4	5 ... n-1	n

Table 2.1: Structure of a Command-APDU [24]

- **Class Field**
  The class field (CLA) is a one byte field. This field contains the class of the APDU. For example, it indicates if the APDU is part of a command chain or if it is encrypted. The CLA field uses a bit mask to indicate the used parameters. If logical channels are supported by the card, the CLA field also indicates the used logical channel. Logical channels, if supported by the smartcard, can be used, to communicate with multiple applications on a smartcard over one physical channel.

- **Instruction Field**
  The instruction field (INS) indicates the command for the card to process. This field is also one byte long. Available commands are defined in ISO 7816-4 [24]. Important commands for electronic passports are described in more detail in this section.

- **Parameter Fields**
  P1 and P2 are each one byte long and contain parameters of the command in the INS field. If there are no parameters needed, P1 and P2 are set to 0.

- **Datalength Field**
  The L_c field is a one byte field, that indicates the length of the data sent in the Command-APDU. There are extended length APDUs which use additional bytes to carry the length.
- **Data field**
  The data field can be up to 256 bytes under normal circumstances. If extended length APDUs are used, it can be up to 65536 bytes long. The data field carries the data sent with the instruction.
- **Length expected**
  The L_e field defines the amount of data that is expected in the response. If this field is 0 it is set to the maximum. This means 256 bytes or 65536 bytes with extended length APDUs.

In the following we discuss some APDUs that are important for our work with the Austrian ePassports.

- **Read Binary** Reads the content of an elementary file on the card. The filesystem of a smartcard and especially the files used in electronic passports are described in Section 2.6.
- **Get Challenge** This command requests a challenge from the card. This challenge may be for example a random number for cryptographic authentication. *Get Challenge* is used for the Basic Access Control (see Section 3.1).
- **Internal Authenticate** Requests the card to compute some authentication data based on the challenge sent from the reader to authenticate the card against the reader. This command can be used to authenticate the card as a whole or just a single application.
- **External Authenticate** This command authenticates the reader against the card based on a challenge received from the *Get Challenge* command. The card responds with a success message.
- **Mutual Authenticate** This command uses the same techniques as the *Internal Authenticate* and the *External Authenticate* command. It uses the challenge from the *Get Challenge* command to authenticate the reader and a challenge from the reader to authenticate the card. This command can also be used to generate a common session key for opening a Secure Messaging channel.
- **Manage Security Environment** This command is used to prepare Secure Messaging and other security commands. Therefore, it supports the four functions: *SET*, *STORE*, *RESTORE* and *ERASE*.
- **General Authenticate** The *General Authenticate* command can be used to authenticate the reader, the card or both. It can not only handle challenge-response mechanisms like the aforementioned authentication commands but other forms like witness-challenge-response triples as well. Therefore, multiple *General Authenticate* commands are exchanged.

Table 2.2 shows the general structure of the Response-APDU. The response from the card consists of the response data if there is any and the status words SW1 and SW2. Each status word is one byte long. If there was an error while processing the Command-APDU the status words contain error codes with more information to the occurred error. If there was no error the status words are set to 0x9000. Table 2.3 and Appendix A show the meaning of the possible statuswords.

Data	SW1	SW2
0 ... n-2	n-1	n

Table 2.2: Structure of a Response-APDU [24]

	SW1-SW2	**Meaning**
Normal processing	’9000’	No further qualification
	’61XX’	SW2 encodes the number of data bytes still available
Warning processing	’62XX’	State of Non-volatile (NV) memory is unchanged
	’63XX’	State of NV memory has changed
Execution error	’64XX’	State of NV memory is unchanged
	’65XX’	State of NV memory has changed
	’66XX’	Security-related issues
Checking error	’6700’	Wrong length; no further indication
	’68XX’	Functions in CLA not supported
	’69XX’	Command not allowed
	’6AXX’	Wrong parameters P1-P2
	’6B00’	Wrong parameters P1-P2
	’6CXX’	Wrong L_e field
	’6D00’	Instruction code not supported
	’6E00’	Class not supported
	’6F00’	No precise diagnosis

Table 2.3: General meaning of the values of SW1-SW2 [24]

2.6 Logical Data Structure (LDS)

According to ISO 7816-4 [24], there are generally two types of files. Dedicated Files (DF) can hold other Dedicated Files or Elementary Files (EF) but can hold no data. The actual data is stored in the Elementary Files. In a classic filesystem, like NTFS for example, Dedicated Files can be compared to folders and Elementary Files are the files holding the data. If there are multiple Dedicated Files with one common parent Dedicated File, then this file is called Master File (MF). There are two types of Elementary Files. Internal Elementary Files contain data used by the card itself. Working Elementary Files can only be used by the outside world, for example through a *Read Binary* command. Figure 2.2 shows an example of how such a structure can look

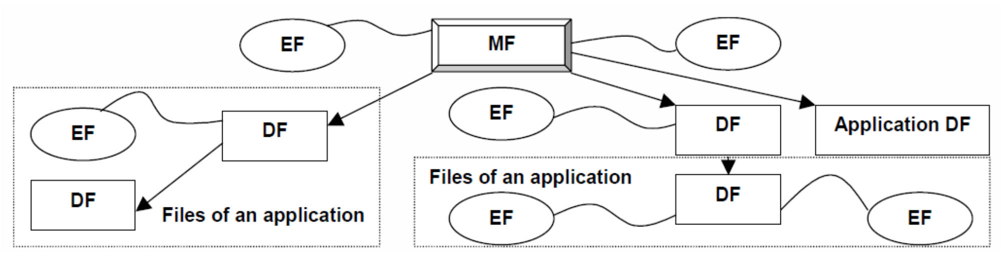


Figure 2.2: Example of a possible structure of a card filesystem [24]

like on a card filesystem.

The Logical Data Structure (LDS) of the electronic passport is divided into Data Groups (DG). These Data Groups hold all necessary data. Each Data Group is stored in an Elementary File called EF.DGx, where the 'x' stands for the number of the Data Group. So DG1 for example is stored in an Elementary File called EF.DG1. Figure 2.3 shows an overview over all Data Groups that are specified in the ICAO 9303. Some of them are mandatory and must be used by the states issuing electronic passports, while some of them are optional. In the following we discuss the Data Groups used in the Austrian electronic passports in more detail. All data elements are stored in a Tag Length Value (TLV) format using the ASN.1 notation [25].

- **DG1**
  DG1 contains the same information as the Machine Readable Zone (MRZ) printed on the data page of the passport. This MRZ contains the name, nationality, date of birth and gender of the document holder, as well as the document type, issuing state or organization, document number and date of expiry of the passport. Optionally additional data about the document holder can be stored in DG1. This is not the case in Austrian passports.

- **DG2**
  The DG2 contains the electronic encoding of the photo printed on the passport. JPEG and JPEG2000 can be used as compression algorithms. The Austrian ePassport contains a JPEG compressed picture.

- **DG3**
  DG3 contains a picture of the fingerprint. This Data Group is only available in second generation ePassports with fingerprints. In Austria, these have been issued since 2009 as described in the analysis of this thesis (see Section 3.7).

- **DG11**
  In DG11, the full name and the place of birth are stored. The name is also stored in DG1, but because of the limited space in the MRZ on the data page, the name might be abbreviated.

- **DG12**
  DG12 contains the name of the issuing authority and the date of issue.

- **DG14**
  DG14 contains the public key for the Chip Authentication protocol described in Section 3.4. This Data Group is only available in Austrian ePassports supporting Extended Access Control (see Section 3.4).

- **DG15**
  In DG15, the public key for the Active Authentication 3.3 is stored. Only Austrian ePassports supporting Active Authentication contain this Data Group.

- EF.SOD
  The EF.SOD contains hash values of all Data Groups available on the passport chip as well as the signature and the DS-Certificate used for Passive Authentication, which is described in more detail in Section 3.2.

- EF.COM
  The EF.COM file contains an Application Identifier (AID), which is not the same as the ISO 7816 AID, the LDS version number and the Unicode version number. The EF.COM file also contains a presence map that indicates the Data Groups that are present on the chip. The EF.COM file is not secured by Passive Authentication, which can lead to the index manipulation attack described in Section 5.5.

2.7 Chip Specifications and Performance

The information about the used chip and the operating system of the second generation Austrian ePassport can be found in [26], as it is certified by the German *Bundesamt für Sicherheit in der Informationstechnik* (BSI). The chip used is the NXP P5CD080 with 80 KB of EEPROM for storage and 6 KB of RAM [27]. The chip operating system is STARCOS 3.3 developed by the German company Giesecke & Devrient[3].

Command	**Response Time**
GetChallenge	29 ms
Mutual Authenticate	166 ms

Table 2.4: Performance of a first generation ePassport

The chip that is actually used in first generation Austrian ePassports is not published and the chip itself gives no indication, which manufacturer or model was used. The same holds true for the used operating system. From [28] we could learn, that NXP Semiconductors[4] (former Philips Semiconductors) produced the first generation ePassports in Austria. At this time the P5CD072 was NXPs ePassport chip. The specification [29] shows that this is a chip with 72 KB

[3] `http://www.gi-de.com`
[4] `http://www.nxp.com`

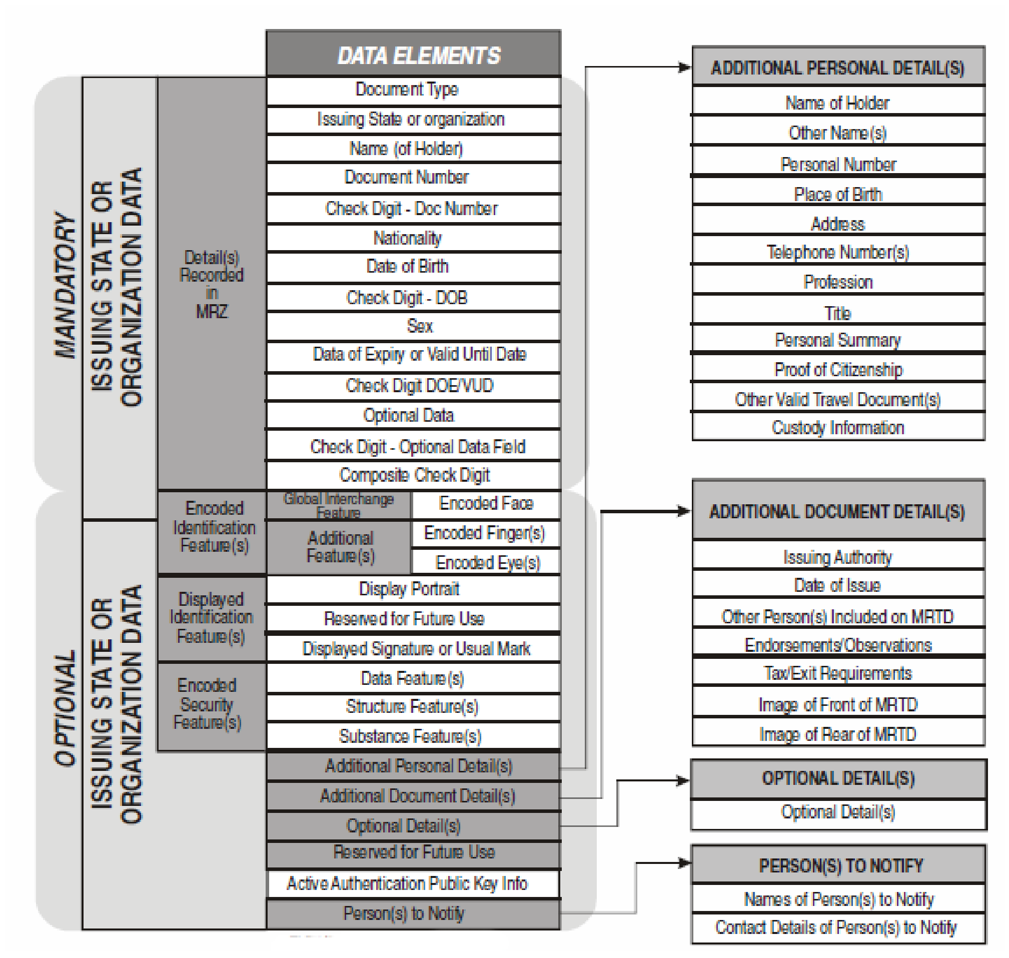


Figure 2.3: Overview over all specified Data Groups [5]

EEPROM for storage and 4.5 KB of RAM.

Table 2.4 and Table 2.5 show the response times of a first and second generation ePassport of the commands necessary for the Basic Access Control (see Section 3.1). As it can be seen in the tables, the response times of the second generation are much higher than the ones of the first generation, in spite of the better hardware in the second generation. One reason for this could be, that the manufacturer deliberately slowed down the cryptographic functions, to make bruteforce attacks more difficult.

Command	**Response Time**
GetChallenge	41 ms
Mutual Authenticate	248 ms

Table 2.5: Performance of a second generation ePassport

CHAPTER 3

Security Features

This chapter first gives a detailed description of the security features specified in the ICAO 9303 documents. It describes mandatory as well as optional features. The second part of the chapter is made up by the analysis of the Austrian implementation. Therein the implementation of the security features is analyzed, with an emphasis on used algorithms and key lengths. Figure 3.1 shows an overview of these security features.

3.1 Basic Access Control (BAC) and Secure Messaging (SM)

The Basic Access Control (BAC) is defined in [5]. It is a security measurement used to prevent skimming and is based on a symmetric key, that can be derived from part of the 'Machine Readable Zone' (MRZ) on the bottom of the data page (see Figure 3.2). As basis for the key, the document number, date of birth and date of expiry and the according check digits are used. The idea behind the Basic Access Control is, that an inspection system can only calculate the access key, when it is able to read the data page of the passport.

Secure Messaging (SM) is also defined in [5] and ensures encryption as well as integrity protection of all transmitted data between the reader and the passport and is based on a symmetric cipher. The session keys for the Secure Messaging are established during the mutual authentication of the Basic Access Control.

Key derivation

The derivation of the needed keys is described using an example where the MRZ looks like the following:

```
P<UTOERIKSSON<<ANNA<MARIA<<<<<<<<<<<<<<<<<<<
L898902C<3UTO6908061F9406236ZE184226B<<<<<14
```

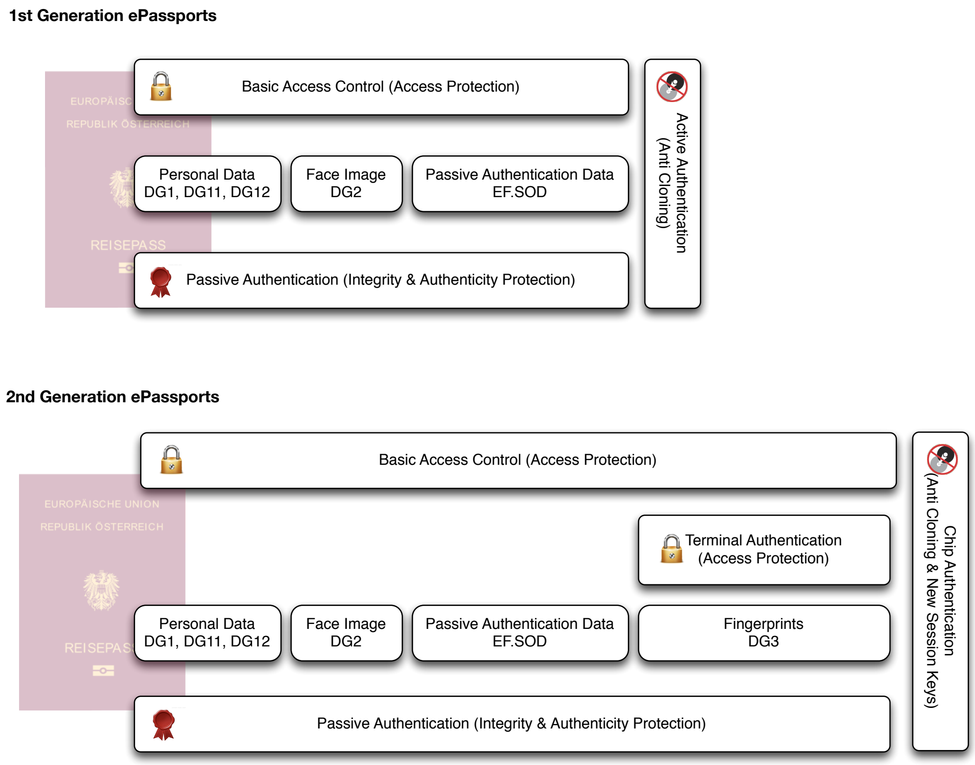

1st Generation ePassports
Basic Access Control (Access Protection)
Personal Data DG1, DG11, DG12
Face Image DG2
Passive Authentication Data EF.SOD
Passive Authentication (Integrity & Authenticity Protection)
Active Authentication (Anti Cloning)
2nd Generation ePassports
Basic Access Control (Access Protection)
Terminal Authentication (Access Protection)
Personal Data DG1, DG11, DG12
Face Image DG2
Passive Authentication Data EF.SOD
Fingerprints DG3
Passive Authentication (Integrity & Authenticity Protection)
Chip Authentication (Anti Cloning & New Session Keys)

Figure 3.1: Overview of the security features used in Austrian ePassports

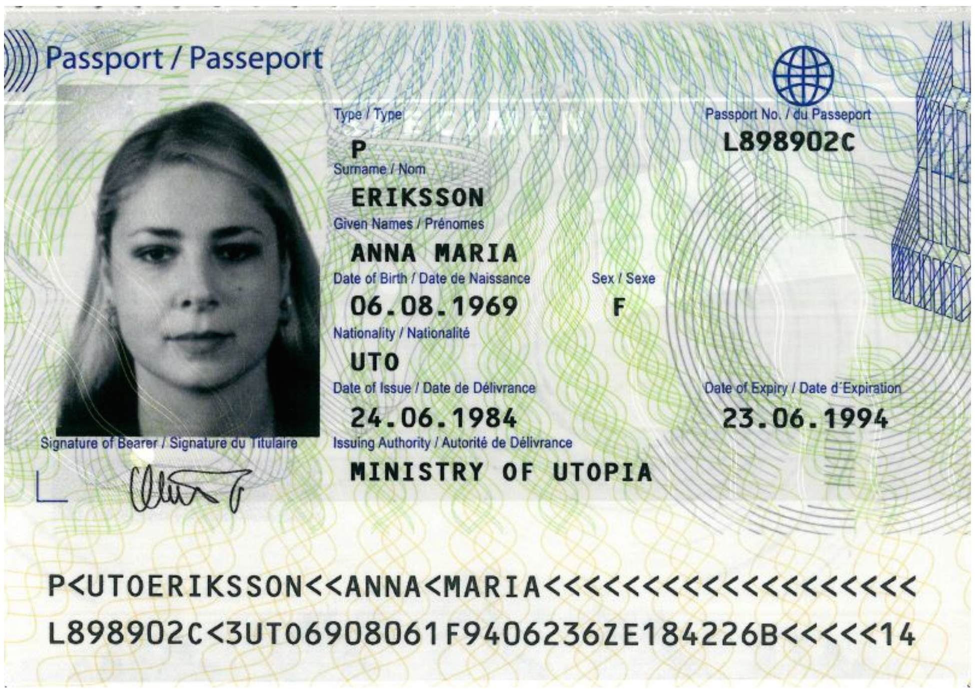
As described before, the key derivation algorithm needs the document number, the date of birth and the date of expiry including the check digits at the end. So the strings needed are:

Document Number:	L898902C<3
Date of Birth:	6908061
Date of Expiry:	9406236
MRZ Information:	L898902C<369080619406236

The first step in the key derivation algorithm is to calculate K_{seed}. This is done by taking the 16 most significant bytes of the SHA-1 hash of the MRZ Information.

$$\begin{aligned} H_{SHA-1}(\text{MRZ Information}) &= \text{239AB9CB282DAF66231DC5A4DF6BFBAEDF477565} \\ K_{seed} &= \text{239AB9CB282DAF66231DC5A4DF6BFBAE} \end{aligned}$$

To be able to derive multiple keys from one key seed, a 32-bit counter (c) is used. To calculate K_{enc}, the key, which later is used for encryption and decryption, K_{seed} is concatenated with this counter, which is set to 0x00000001. From this string, the SHA-1 hash is calculated and the 16 most significant bytes are used build K_{enc}.


Passport / Passeport

Type / Type
P

Surname / Nom
ERIKSSON

Given Names / Prénomes
ANNA MARIA

Date of Birth / Date de Naissance
06.08.1969

Sex / Sexe
F

Nationality / Nationalité
UTO

Date of Issue / Date de Délivrance
24.06.1984

Issuing Authority / Autorité de Délivrance
MINISTRY OF UTOPIA

Passport No. / du Passeport
L898902C

Date of Expiry / Date d'Expiration
23.06.1994

Signature of Bearer / Signature du Titulaire

```
P<UTOERIKSSON<<ANNA<MARIA<<<<<<<<<<<<<<<<<<<
L898902C<3UTO6908061F9406236ZE184226B<<<<<14
```

Figure 3.2: Example of a passport's datapage

$$\begin{aligned} K_{seed} \,\&\, c &= \text{239AB9CB282DAF66231DC5A4DF6BFBAE0000001} \\ H_{SHA-1}(K_{seed} \,\&\, c) &= \text{AB94FCEDF2664EDFB9B291F85D7F77F27F2F4A9D} \\ K_{enc} &= \text{AB94FCEDF2664EDFB9B291F85D7F77F2} \end{aligned}$$

To guarantee the integrity of messages while communicating, so called Message Authentication Codes (MAC) are used. To calculate these MACs, a separate key (K_{mac}) is used. For the calculation of K_{mac}, the counter (c) is set to 0x00000002 and concatenated with K_{seed}. Like before, the SHA-1 hash from this string is calculated. The 16 most significant byte from this hash form the key K_{mac} .

$$\begin{aligned} K_{seed} \,\&\, c &= \text{239AB9CB282DAF66231DC5A4DF6BFBAE0000002} \\ H_{SHA-1}(K_{seed} \,\&\, c) &= \text{7862D9ECE03C1BCD4D77089DCF131442814EA70A} \\ K_{mac} &= \text{7862D9ECE03C1BCD4D77089DCF13144} \end{aligned}$$

According to the specification, the 8th bit of each byte in a DES key is used for parity. To form a valid key, each byte has to have an odd parity. Therefore, the resulting keys should be checked to adhere to this specification. If necessary, bytes with an even parity need to be adjusted. After this final adjustment, the keys look like the following:

$$\begin{aligned} K_{enc} &= \text{AB94FDECF2674FDF B9B391F85D7F76F2} \\ K_{mac} &= \text{7962D9ECE03D1ACD 4C76089DCE131543} \end{aligned}$$

Mutual Authentication and Session Key Establishment

The idea behind the mutual authentication is, that the inspection system authenticates against the chip, as well as the chip against the inspection system, whereas even a rogue chip can easily calculate the needed keys for the authentication. Therefore other features, like Active Authentication and Chip Authentication have been specified for authenticating the chip. One result of the mutual authentication, besides the chip granting access to the stored data, both parties calculate a common session key. The message flow of the mutual authentication can be seen in Figure 3.3. The first step is to send the *Get Challenge* command to the passport. The answer to this is an 8-byte long random nonce ($RND.ICC$). The reader then calculates two random values. One is an 8-byte random nonce ($RND.IFD$) and the second one is a 16-byte random (K_{IFD}), which is later used for the key establishment. $RND.IFD$, $RND.ICC$ and K_{IFD} are then concatenated to the string S. In the worked example the values look as following:

$$\begin{aligned} RND.ICC &= \text{4608F91988702212} \\ RND.IFD &= \text{781723860C06C226} \\ K_{IFD} &= \text{0B795240CB7049B01C19B33E32804F0B} \\ S &= RND.ICC \;\&\; RND.IFD \;\&\; K_{IFD} \end{aligned}$$

The reader encrypts S with K_{enc} using 3DES in CBC mode and gets E_{IFD}. It then obtains M_{IFD} by calculating a 3DES-MAC using K_{mac}. The concatenated strings (E_{IFD} & M_{IFD}) are sent to the passport using the *Mutual Authentication* command.

The passport checks the MAC and decrypts the ciphertext. In the next step it calculates K_{ICC}, which is used for establishing the common session key. It then answers with $RND.ICC$, $RND.IFD$ and K_{ICC}.

The inspection system then checks the MAC of the message. If it is correct, the inspection system decrypts the ciphertext and compares $RND.IFD$ with the value it chose before. To calculate the session key both parties calculate a key seed by XORing K_{IFD} and K_{ICC}. From the key seed they calculate the session keys KS_{enc} and KS_{mac} the same way K_{enc} and K_{mac} from K_{seed} in the previous section.

Secure Messaging

To protect the confidentiality and integrity of the transport channel between the passport and the reader, Secure Messaging is used. To protect a Command APDU using Secure Messaging, the CLA field is exchanged with ’0C’, indicating, that the following APDU is a protected APDU. The rest of the command header (INS, P1, P2) is not altered. Following the header, the length of the new protected APDU (L_c') is appended. The next four bytes are used for padding and are therefore set to ’80’ ’00’ ’00’ ’00’. In the next step the sent data is first padded according to ISO 9797-1 method 2, encrypted with KS_{enc} and then prepended with ’87’ L ’01’ whereas L indicates the length of the cryptogram. In the last step, the L_e field is prepended with ’97’ ’01’ and appended to the rest of the APDU. To protect the integrity of the APDU, the newly built

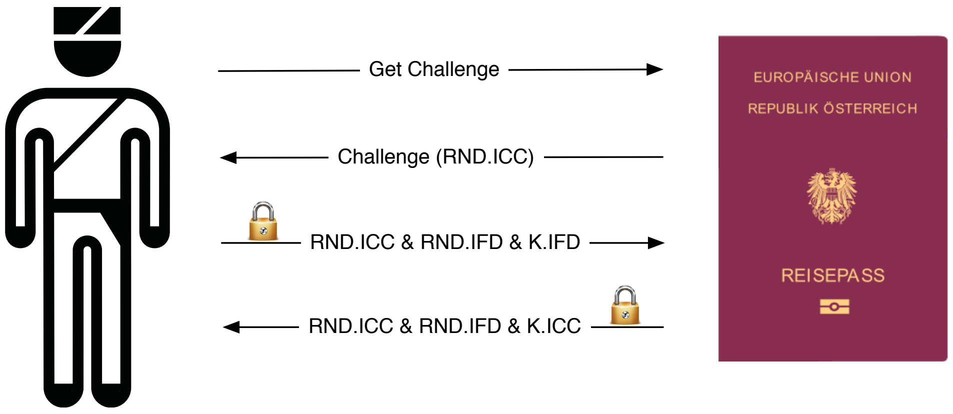


Figure 3.3: Message flow of the mutual authentication

APDU is padded and then a MAC is calculated using KS_{mac}. The MAC is then appended to the APDU with the string '8E' '08' CC '00', where CC is the calculated MAC.

As we can also see, is, that only the sent data is encrypted. The command header is sent as cleartext over the transport channel. An attacker eavesdropping on the communication could, at least, find out what command is sent to the chip.

Similar to the Command APDU, also the Response APDU is protected. In the first step, the data is encrypted using KS_{enc} and prepended with '87' L '01', whereas L indicates the length of the encrypted data. In the next step, '99' '02' and the two status words are added to the response. Finally the MAC is calculated in the same way as with the Command APDU. Therefore the built APDU is padded and then the MAC using KS_{mac} is calculated and appended to the APDU with the string '8E' '08' CC '00', where CC is again the calculated MAC. Like in the command APDU, the status word is not encrypted, which could lead to information disclosure. Both processes are shown in Figure 3.4.

3.2 Passive Authentication (PA)

The Passive Authentication (PA) is used to protect the authenticity and integrity of the data stored on the chip. Therefore, the EF.SOD contains hash values of all the data groups on the chip and a digital signature of the hash values. The digital signature is created using the Document Signer (DS). The DS certificate can be stored on the chip in the EF.SOD, which is the case in the Austrian implementation, and is itself digitally signed by the Country Signer (CS). The Austrian Country Signer Certificate (CSCA) can be obtained from the homepage of the ministry of the interior[1]. Austria also participates in the ICAO Public Key Directory (PKD) (Section 3.6).

[1]`http://www.bmi.gv.at/cms/bmi_service/csca/`

```
SEQUENCE {
  OBJECT IDENTIFIER mRTDSignatureData (2 23 136 1 1 1) [0] {
    OCTET STRING, encapsulates {
      SEQUENCE {
        INTEGER 0
        SEQUENCE {
          OBJECT IDENTIFIER sha-256 (2 16 840 1 101 3 4 2 1)
          NULL
        }
        SEQUENCE {
          SEQUENCE {
            INTEGER 1
            OCTET STRING
            3A 35 85 51 06 AD 7A F9 A1 09 9D 9F 68 36 EF 11
            60 29 4B D8 BF 28 03 1F F8 44 80 2C 1B 9F 53 48
          }
          SEQUENCE {
            INTEGER 2
            OCTET STRING
            81 0E FE 98 AB 5D C9 DF 51 37 3E A5 AA 48 9F 44
            AF 6E 66 EF 71 24 27 7E CA E3 76 B0 D5 BB B1 1B
          }
          SEQUENCE {
            INTEGER 3
            OCTET STRING
            DA 2A DF DE F4 76 DE 8D 96 8E DB FB CA EB 43 2E
            82 7C F9 E6 7B 1C B8 3B 30 AF 70 25 BA 26 77 DB
          }
          SEQUENCE {
            INTEGER 11
            OCTET STRING
            62 B2 C8 B2 9D E6 3C 2B 3C 38 2F 4E 68 6A 5E D5
            EB 95 EB 1C 74 79 DB 23 7A 77 B2 C8 1E 6A 96 9C
          }
          SEQUENCE {
            INTEGER 12
            OCTET STRING
            1A 44 5D 3E 1E C8 90 CF 2F D3 06 6C 05 68 97 8B
            C9 97 A1 EA 7E AC 3A 38 37 B6 72 FE C4 65 1D F4
          }
          SEQUENCE {
            INTEGER 14
            OCTET STRING
            79 06 F1 09 87 F2 9A B7 0A 73 B8 B8 3F 01 94 8C
            CC 8A 61 3E AC E6 C8 9A E6 00 F3 6B A7 46 52 45
          }
        }
      }
    }
  }
}
```

Listing 3.1: Excerpt of the hash value part of an EF.SOD file

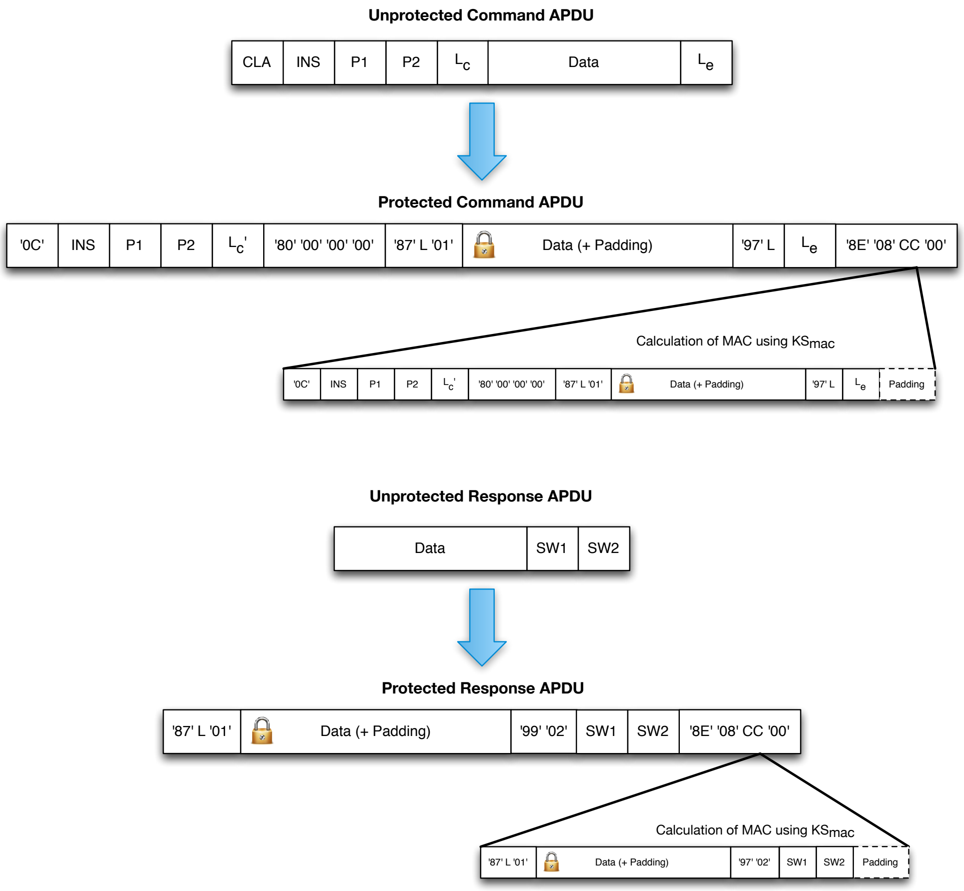


Figure 3.4: Encryption of a Command APDU and a Response APDU using Secure Messaging

Listing 3.1 shows the hash value part of an EF.SOD file. It can be seen, that there is a hash value for each data group on the chip. We can also see, that there is no hash value for the EF.COM, which is often used as an index. If an inspection system relies solely on the unprotected EF.COM as an index, it can lead to the index manipulation attack described in Section 5.5.

3.3 Active Authentication (AA)

Active Authentication (AA) protects the chip against cloning. To prove the chip's authenticity, a challenge-response algorithm is used, as it can be seein in Figure 3.5. The inspection system sends an 8-byte nonce ($RND.IFD$) to the passport using the *Internal Authenticate* command. The passport calculates a 106-byte nonce (M_1) whereas the length of this nonce (L_{M_1}) is calculated from the used key length (k) and the length of the used hash (L_h).

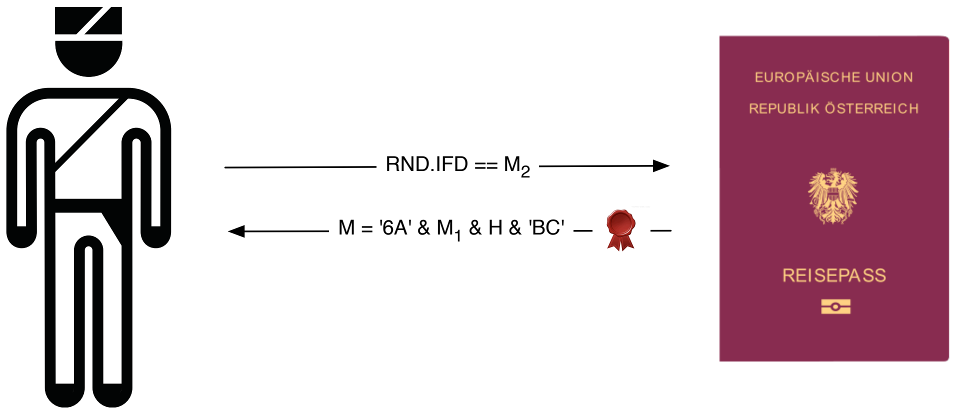


Figure 3.5: Message flow of the Active Authentication protocol

$$\begin{aligned} c &= k - L_h - 8 - 4 = 1024 - 160 - 8 - 4 = 852bit \\ L_{M_1} &= c - 4 = 852 - 4 = 848bit = 106byte \end{aligned}$$

In the next step, the passport concatenates $RND.IFD$ (M_2) and the generated nonce (M_1) to build M. It then calculates the SHA-1 hash of M (H) and constructs the so-called *message representative* (F) by concatenating $'6A'$, M_1, H and $'BC'$ where $'BC'$ indicates the used hash function. In the last step, the passport encrypts the message representative using its private key and sends it back to the inspection system.

The inspection system reads the passport's public key stored in DG15 and verifies its authenticity using Passive Authentication (Section 3.2). It then decrypts the received message using this public key. In the next step it extracts the calculated hash (H) and the message representative (M_1). After that it concatenates M_1 with the known M_2, calculates the SHA-1 hash and compares it to the extracted H. If they match, Active Authentication was successful. Figure 3.5 shows the message flow of the Active Authentication.

The security of the Active Authentication heavily relies on the security of the Passive Authentication and the non exportability of the private key stored on the chip. If an attack is able to export the private key, or change the public key in DG15 without the inspection system noticing it, the Active Authentication is useless.

The described challenge-response protocol used for Active Authentication is defined in ISO9796-2 [30].

3.4 Extended Access Control (EAC)

The Extended Access Control (EAC) was introduced by the German 'Bundesamt für Sicherheit in der Informationstechnik' (BSI) in 2006 in the first version of the TR-03110 [31]. It consists of two parts. One part is the Terminal Authentication, which was intended as an access control mechanism for additional biometric features (e.g. fingerprints) stored on the chip, and the other part is the Chip Authentication, which is a new protocol for checking the authenticity of the ePassport's chip. Chip Authentication can be implemented instead of or in addition to the Active Authentication (Section 3.3). The specifications can be found in [31]. For more technical issues, like APDU sequences and used data groups, a worked example for the Extended Access Control by the BSI was used [32].

Chip Authentication (CA)

The Chip Authentication (CA) is based on a Diffie-Hellman key agreement protocol [33]. It provides authentication of the chip and establishes a more secure Secure Messaging channel by providing stronger session keys.

As the first step, the reader retrieves the public key (PK_{ICC}) and the domain parameters (D_{ICC}) from DG14. The reader then uses the domain parameters to choose its own private ($\widetilde{SK_{IFD}}$) and public ($\widetilde{PK_{IFD}}$) keys. In the next step it prepares the chip using the *Manage Security Environment* command. This step is used, if there are multiple key pairs saved on the chip, to select the right key id and to prepare the chip for the next command. Using the *General Authenticate* command, the reader sends the chosen public key to the chip. Both parties use their private key, the opposite's public key and the domain parameters to calculate the new session key by using the Diffie-Hellman key agreement function (KA).

$$K = \text{KA}(\widetilde{SK_{IFD}}, PK_{ICC}, D_{ICC})$$

This new session key is then used to open a new Secure Messaging channel. The authenticity of the chip can be guaranteed by checking the public key using Passive Authentication (see Section 3.2). Figure 3.6 shows the message flow of the Chip Authentication protocol.

Terminal Authentication (TA)

The Terminal Authentication (TA) enables the chip to verify that the inspection system is authorized to access certain data groups. The Terminal Authentication is based on a two move challenge-response protocol. Before the inspection system can authenticate itself using the Terminal Authentication it must verify the chip using the Chip Authentication.

The Terminal Authentication consists of two major parts. A preparation step and an authentication step. Both steps are visualized in Figure 3.7. In the preparation step, the inspection

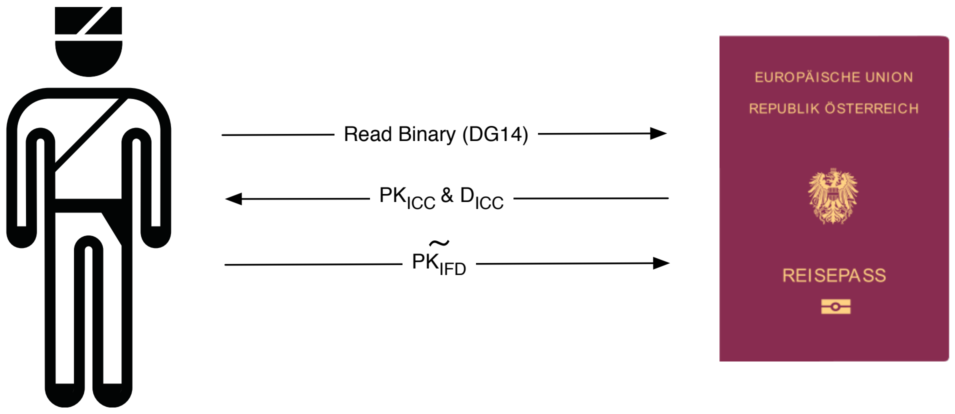


Figure 3.6: Message flow of the Chip Authentication protocol

system sends a certificate chain to the chip using the *Manage Security Environment* command. The certificate chain starts with a certificate that can be verified against the Country Verifying CA (CVCA) and ends with the terminal's own certificate. The possible certificates in the certificate chain sent to the chip can be found in Section 3.6. Every certificate is sent using the *Manage Security Environment* and verified by the chip after the reader sends the *Perform Security Operation: Verify Certificate*.

In the authentication step, the inspection system sends the *General Authenticate* command to the chip, which responds with a random value (r_{ICC}). The inspection system signs this value along with the passport number (ID_{ICC}) and a compressed version of the reader's public key from the Chip Authentication protocol ($Comp(PK_{IFD})$). The chip verifies the signature using the reader's public key and grants access to the sensitive data stored on the chip. The data an inspection system has access to, is stored in the so called Access Map inside the inspection system's terminal certificate.

To be able to verify a terminal certificate that was issued by a newer CVCA than active when the chip was issued, the terminal has to present the chip a so-called link certificate. This link certificate links the new CVCA to the one, that is stored on the chip. If the chip can successfully verify the new CVCA, it is saved as a so-called trust point in the chip's secure memory. The next time the chip is presented with the new CVCA it uses this trust point to verify the DV and terminal certificate.

To verify the certificates in the chain presented by the reader, the chip has to know if one of them are already expired. To do this it needs to have an accurate time source. Due to the fact, that a chip is not always powered, it cannot make use of an internal clock. To compensate this problem, the chip uses the issuance dates of presented terminal certificates. If it can verify the validity of a terminal certificate and the issuance date is later than its own current date, it

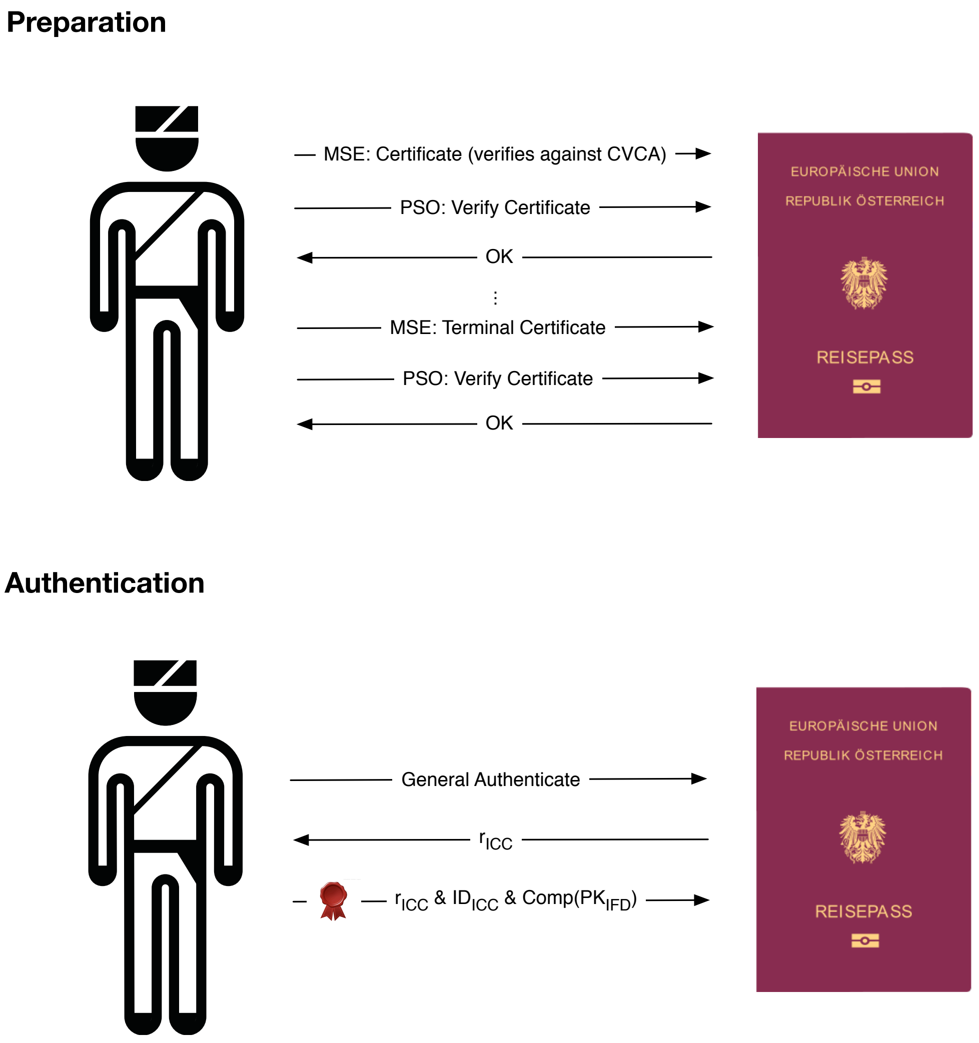


Figure 3.7: Message flow of the Terminal Authentication protocol

stores this date as current date. This means that a chip never has a really accurate time source. Although this could lead to a security problem shown in Section 5.6, this is the only feasible method, as it is not possible for the chip to be constantly connected to an accurate time source.

3.5 Password Authentication Connection Establishment (PACE)

Due to the weaknesses in the Basic Access Control a new way of access control to protect the data on the ePassport chip was developed. The Password Authentication Connection Establishment (PACE), in contrast to the Basic Access Control, uses an asymmetric cipher based on the Diffie-Hellman (DH) key exchange. PACE is defined by the ICAO in [34].

PACE calculates strong session keys using a password. This password can either consist of parts of the MRZ, as it is used in the Basic Access Control, or it can be the Card Access Number (CAN). This CAN is an arbitrary number printed somewhere inside the passport. The exact location of the CAN is not specified. If PACE is implemented by the ePassport chip, it has to at least support the MRZ password. The CAN is optional, but has the advantage that it is shorter than the MRZ and can so be typed in easier manually.

The message flow of PACE is shown in Figure 3.8. The protocol can, like the Terminal Authentication protocol, be divided into a preparation and an authentication part. To perform the authentication and key exchange using the PACE protocol, the reader has to read the EF.CardAccess. This file contains the supported protocols and domain parameters to perform the protocol. If the chip supports multiple algorithms (for example with different key lengths) multiple domain parameters are stored on the chip.

After retrieving the parameters for the PACE protocol from EF.CardAccess, the reader derives a key (K_π) from the password, using the Key Derivation Function (KDF). The used KDF is also stored in the EF.CardAccess file on the chip and depends on the used algorithm, but is always some hash function, like SHA-256 or something similar. The reader sends a *Manage Security Environment* command to the chip, selecting one of the cipher suites from the EF.CardAccess file and a password type, which is either MRZ or CAN. Following this, the chip generates a 16-byte nonce (s) and encrypts it using K_π with the symmetric cipher defined in the EF.CardAccess. The reader then requests the encrypted nonce (z) and the static domain parameters (D_{ICC}) using the *General Authenticate* command. After retrieving s, by decrypting z, the reader, as well as the chip, calculates ephemeral domain parameters from D_{ICC} and s. The chip and reader both generate a secret and a public key based on the ephemeral domain parameters (PK_{ICC}, SK_{ICC}, PK_{IFD}, SK_{IFD}) and exchange the public keys. After the exchange, both entities calculate the key seed (K) and derive a MAC key (K_{mac}) and an encryption key (K_{enc}) from this key seed. To verify the successful execution of PACE, the reader sends PK_{ICC} and a MAC calculated with K_{mac} using the *Mutual Authenticate* command. As response the chip sends PK_{IFD} including a MAC also calculated with K_{mac} back to the reader. If both parties are able to verify the MAC, the PACE protocol was successfully executed.

3.6 ICAO Public Key Infrastructure

The general structure of an ICAO Public Key Infrastructure (PKI) is split into two parts. One part is defined in [5] and is used for the Passive Authentication. This tree starts with the CSCA as root. The CSCA certificate is self-signed and is therefore independent. To make the processes of certificate renewals, revocations and exchange between states easier, so-called link certificates can be used. These link certificates contain the public key of the new CSCA and are signed by the old CSCA. This makes sure, that the link to the old CSCA can be verified [35].

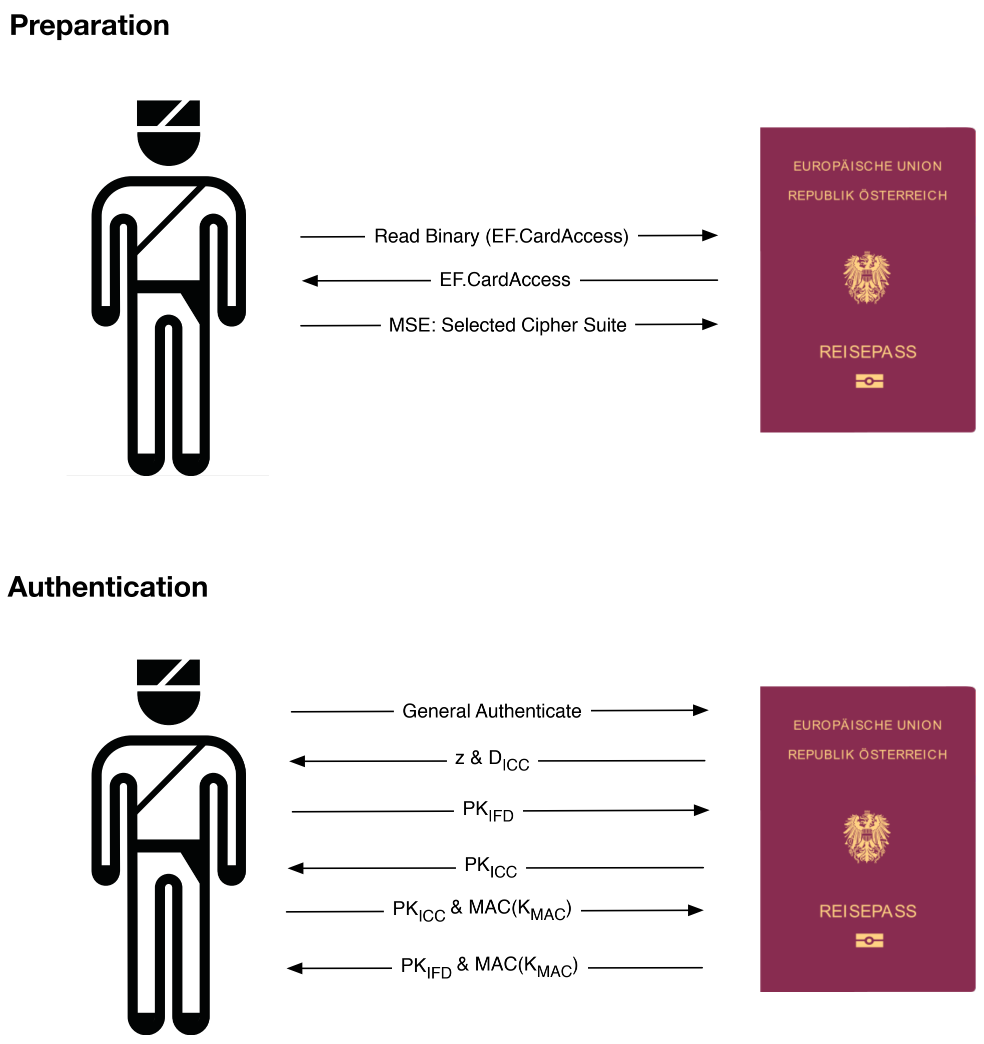


Figure 3.8: Message flow of the PACE protocol, including preparation and authentication part

The EF.SOD containing the hash values of all used data groups is signed by the DS (Section 3.2). The DS certificate is signed by the CSCA and can therefore be verified against it. The DS certificate itself can be stored on the chip in the EF.SOD. If this is the case, like it is in Austria, it must be read by a reader prior to the execution of the Passive Authentication.

In [5], it is stated that states should exchange CSCA certificates and DS certificates on a bilateral basis. As this means a lot of overhead for the passport issuing authorities in the different countries, the ICAO started the ICAO Public Key Directory (PKD). The PKD enables member states to upload their DS certificates and revocation lists. If a member state wants to verify an ePassport of another member state, all it has to do is download the certificate and the revocation

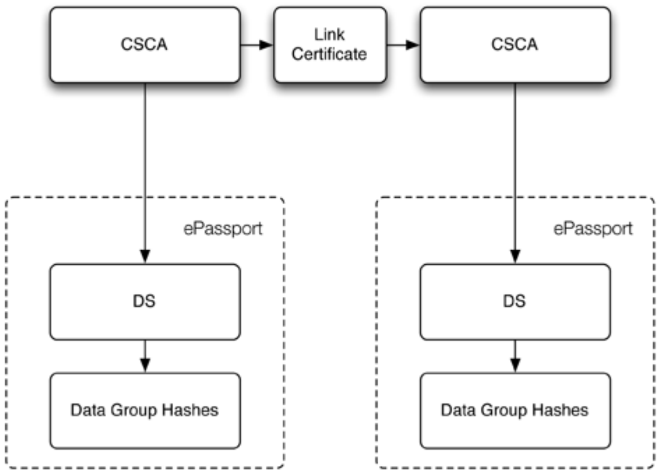


Figure 3.9: Illustration of the ICAO PKI for Passive Authentication

list from the PKD.

At the moment of writing, there are 37 participants in the PKD out of 100 states issuing ePassports. Looking at the total number of issued ePassports it can be seen, that 74% of all issued ePassport have been issued by PKD member states [36].

Besides the ICAO PKD the German BSI regularly publishes a master list with CSCA certificates and link certificates. At the time of writing this list contains certificates of 54 countries. Also other countries started to generate their own master lists. These master lists can also be uploaded to the ICAO PKD.

Figure 3.9 shows an illustration of an ICAO PKI for the Passive Authentication with two CSCAs and a link certificate.

The second part of the ICAO PKI is used for the Extended Access Control and is specified in [31] and [37]. At the root of the EAC PKI is the CVCA. This is the keypair held by the issuing country. The CVCA signs the DV certificates which are held for example by foreign authorities that have been authorized to read the Extended Access Control protected data groups of the CVCA holder's ePassports. The DVCA itself can issue the terminal certificates which are deployed to the inspection systems actually accessing the ePassports. The issuing country can restrict access to specific data groups when issuing DV certificates. The terminal can then have at most the same or fewer access rights than the DVCA. For example if country A grants country B the right only to read DG3 (i.e. the stored fingerprints) but not DG4 (i.e. the stored iris), country A issues DV certificates with these restrictions to country B. Therefore, country B cannot issues terminal certificates, from these DV certificates, granting access to DG4 on country A's ePassports.

As stated in [37], EU member states must provide a Single Point of Contact (SPOC) for all matters concerning the Extended Access Control key management. This SPOC can for example be implemented as a Web Service and is defined in the Czech Standard CSN 369791:2009 [38]. Table 3.1 shows the messages specified in [38] for SPOCs.

Message	**Intended Use**
RequestCertificate	This message is used by a SPOC for requesting the generation of a new certificate for one of its DV from a foreign CVCA.
SendCertificates	SPOC sends the new certificate or certificate chain to the requesting SPOC.
GetCACertificates	This message is sent by a SPOC to a foreign SPOC in order to get all valid CVCA certificates (link certificates and selfsigned certificates) of that country.
GeneralMessage	This message is sent by a SPOC to a foreign SPOC in order to send notification or other general text human readable message.

Table 3.1: Messages specified in [38] for the Single Point of Contact (SPOC)

Figure 3.10 shows an illustration of the ICAO PKI for the Extended Access Control. Here, on the root of the tree, are the CVCAs of Country A and Country B. Both countries have multiple DV certificates in their own country. These DV certificates can be issued to, for example different borders. So each airport, seaport or land border in Country A has it's own DV certificate. This certificate is then used to issue terminal certificates for the inspection systems at this border. Terminal certificates should have a very short validity period, so that the revocation of such a certificate is easier. Besides DV certificates in their own country, Country A also issues DV certificates for Country B. This enables them to issue terminal certificates, that are able to read the data groups, protected by the Extended Access Control, from Country A's passports. Also Country B issues DVs to Country A, so that they are able to access these data groups on Country B's passports.

3.7 Implementation of Security Features in Austrian ePassports

The following section will show how the above security features are implemented in different versions of Austrian ePassports. To gather the underlying data we collected samples of Austrian ePassports with the issuing date between 2006 (introduction of the ePassport in Austria) and now. The issuing dates as well as all necessary data of all collected samples can be found in

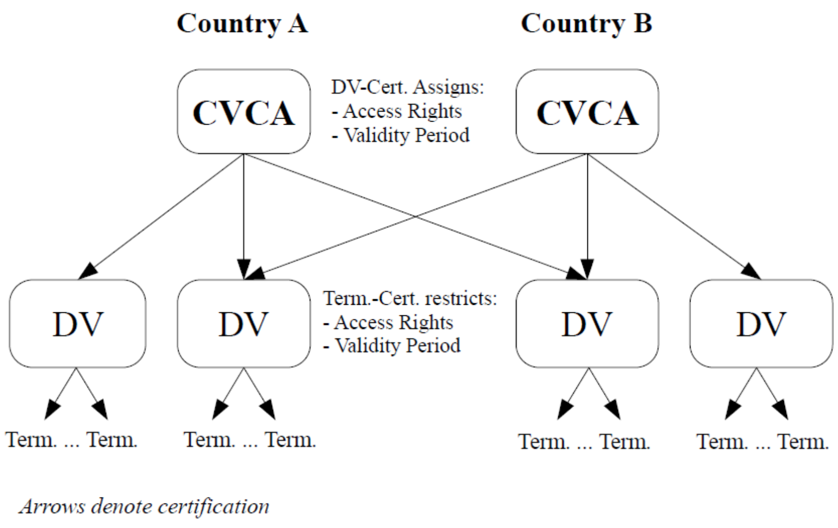


Figure 3.10: Illustration of the ICAO PKI for Extended Access Control [37]

Appendix B. The data we were especially interested in were the used algorithms, key lengths and validity length. We also tried to find significant alterations like the change of algorithms, key lengths or the introduction of new security features.

Basic Access Control

Although [5] states the use of the Basic Access Control as optional, all analyzed passports implemented it. Due to the fact that [5] gives no room for interpretation concerning the implementation of the Basic Access Control, all analyzed passports implemented it the way it was described in the standard.

Passive Authentication

The Passive Authentication depends on two certificates (CSCA and DS). Only the DS certificate is stored on the chip. The Austrian CSCA certificate should be downloaded either from the homepage of the Austrian ministry of the interior (see Section 3.2), the ICAO PKD or the German master list to ensure the authenticity of the stored certificate. Austria has two active CSCA certificates at the moment. The details of these certificates can be found in Table 3.2.

We found that the Austrian CSCA certificate is valid for around 15 years with a private key usage of about 5 years. This is what we expected considering that a standard passport is valid for 10 years in Austria [39]. In 2011 the key length was updated from 3072 bits to 4096 bits. Both key lengths are considered secure at the moment by the National Institute for Standards

Name	Valid From	Valid To	Public Key Algorithm	Key Length
CSCA-AUSTRIA	2006-06-09	2021-09-12	RSA	3072 bits
CSCA-AUSTRIA	2011-04-15	2026-07-19	RSA	4096 bits

Table 3.2: Austrian CSCA certificates details

and Technology (NIST) [40].

Name	Valid From	Valid To	Public Key Algorithm	Key Length
DS-AUSTRIA	2006-06-09	2016-09-12	RSA	2048 bits
DS-AUSTRIA	2006-07-09	2016-12-12	RSA	2048 bits
DS-AUSTRIA	2007-03-02	2017-06-05	RSA	2048 bits
DS-AUSTRIA	2007-06-01	2017-09-04	RSA	2048 bits
DS-AUSTRIA	2007-08-30	2017-12-04	RSA	2048 bits
DS-AUSTRIA	2008-06-04	2018-09-08	RSA	2048 bits
DS-AUSTRIA	2008-12-02	2019-03-09	RSA	2048 bits
DS-AUSTRIA	2009-03-02	2019-06-06	RSA	2048 bits
DS-AUSTRIA	2009-06-03	2019-09-07	RSA	2048 bits
DS-AUSTRIA	2009-09-01	2019-12-07	RSA	2048 bits
DS-AUSTRIA	2010-03-03	2020-06-06	RSA	2048 bits
DS-AUSTRIA-ePass	2011-08-24	2021-11-27	RSA	3072 bits
DS-AUSTRIA-ePass	2012-02-16	2022-05-12	RSA	3072 bits
DS-AUSTRIA-ePass	2013-05-14	2023-08-18	RSA	3072 bits

Table 3.3: Austrian DS certificates details

Table 3.3 shows the DS certificates of the collected passports. We found that the validity of a DS certificate is 10 years and 95 days, which leaves a private key usage period of a maximum of 95 days with a passport validity of 10 years. This is, because the certificate has to be valid at least as long as the passport. So if the validity of the certificate is 10 years and 95 days and the private key was used 100 days for example, the passport would be valid five days longer than the certificate. Until 2011, the key length of the DS certificate was 2048 bits. After the CSCA upgrade to 4096 bits, the DS certificate was upgraded to a 3072 bits key length. Both key lengths are considered secure according to NIST [40].

As a hashing algorithm for both, the signature and the representation of the data groups in the EF.SOD SHA-256 [41] was used. According to NIST [40] the SHA-256 algorithm is considered secure at the moment.

Active Authentication

All passports between 2006 and 2009 were equipped with Active Authentication as a cloning protection. In 2009 two fingerprints were integrated into the Austrian passports and the Extended

Access Control was used to protect these data groups. At the same time Extended Access Control was introduced and the Active Authentication was replaced by Chip Authentication. All passports in the specified time frame implemented Active Authentication with RSA and a key length of 1024 bits which is not recommended to use in new applications [40]. Although Active Authentication is not used in current passport, there are still passports with Active Authentication in the field till 2019. This means that if the Active Authentication key is broken, a reader might not recognize a cloned chip if it is relying on Active Authentication. There are already practical attacks on this signature algorithm published in [42].

Extended Access Control

With the introduction of fingerprints in Austrian passports, Extended Access Control was used to specially protect the EF.DG3 that is used to hold the image of the two fingerprints. To make use of Terminal Authentication, which is used to protect these data groups, Chip Authentication has to be used, too.

Chip Authentication

The Chip Authentication protocol uses a static Diffie-Hellman key agreement protocol where the public key and the domain parameters are stored in DG14. The BSI specifies two possible algorithms for the Chip Authentication protocol [31]. Either a Diffie-Hellman algorithm based on elliptic curves (ECDH) or a standard Diffie-Hellman algorithm (DH) is used. All analyzed passports implementing Chip Authentication made use of the ECDH protocol. Table 3.4 shows the two different curves that have been used since 2009. In 2012 the size of the curve's finite field has been upgraded from 224 bits to 256 bits. The curves in use are the brainpoolP224r1 and the brainpoolP256r1 both specified in [43]. We could find this information by extracting the domain parameter information from DG14 and comparing the result with well-known and often used curves. Both sizes are sufficient for the validity period according to NIST [40].

Date	**Size**	**Curve ID**
2009 - 2012	224 bits	brainpoolP224r1
20120 - now	256 bits	brainpoolP256r1

Table 3.4: Elliptic curve parameters for chip authentication

Terminal Authentication

The Austrian CVCA certificate is not published and cannot be read from the passports, so we could not find the algorithms used for the Terminal Authentication. We could find from the certificate reference in the EF.CVCA that in 2012 a new CVCA certificate was created. This means that the private key of a CVCA is used for about three years. Because of the missing certificate itself, we can not make any assumptions of how long the CVCA certificates are valid.

Password Authentication Connection Establishment

The Austrian ePassport does not implement PACE at the moment. In 2011, the European Commission decided that all member states of the European Union have to implement the Supplemental Access Control (SAC) including PACE by December 31, 2014 [44]. So PACE will be available in Austrian ePassports by 2015. As the decision says nothing about the implementation details, we cannot say at the moment, which algorithms and elliptic curves exactly will be used for the Austrian ePassport.

3.8 Comparison to other countries

In [45], the European Frontex Agency, as part of a study, analyzed the security features used in different passport implementations. The analyzed passports were from the United Kingdom, Germany, Finland, Sweden, Spain, Netherlands, Poland and Russia. All of the samples supported Basic Access Control, which is an optional security feature, according to ICAO 9303, but mandatory by EU regulations, and Passive Authentication, which is mandatory by the ICAO 9303 specifications. Only Germany, Netherlands and Finland implemented the Extended Access Control. Sweden implemented at least Active Authentication as a cloning protection. Finland and the Netherlands implemented Active Authentication alongside the Extended Access Control. Frontex also took a closer look at the algorithms used in Passive Authentication. Germany, Spain and Russia still use SHA-1 as a hashing algorithm. Although SHA-1 is not considered broken, it is not recommended anymore for long term use [40]. All other countries use the SHA-256 hashing algorithm. Table 3.5 shows an overview of the found implementation details.

State	BAC	PA	AA	EAC	PA algorithm	EAC algorithm
AT	●	●	-	●	sha256WithRSAEncryption	ECDH
DE	●	●	-	●	ecdsa-with-sha1	ECDH
UK	●	●	-	-	sha256WithRSAEncryption	-
FI	●	●	●	●	sha256WithRSAEncryption	ECDH
SE	●	●	●	-	id-RSASSA-PSS and sha256	-
ES	●	●	-	-	sha1WithRSAEncryption	-
NL	●	●	●	●	sha256WithRSAEncryption	ECDH
PL	●	●	-	-	sha256WithRSAEncryption	-
RU	●	●	-	-	ecdsa-with-sha1	-

Table 3.5: Comparison of different ePassport implementations [45]

CHAPTER 4

Attack Vectors

This chapter discusses the possible attack vectors on electronic passports. It enumerates and explains possible attacks and gives an understanding of the risks that can result from these attack vectors if they are exploited. Furthermore, it shows which of the described security features are used to prevent the exploitation of an attack vector.

4.1 Eavesdropping and Skimming

Skimming is an attack where an illicit reader accesses the data on the passport chip without the bearer of the passport knowing. Skimming and eavesdropping are especially concerning because of their privacy implications. If an attacker can read a passport's data without the bearer knowing, one could easily create a database of people in places with a high density of people carrying their passport, for example airports or train stations. Due to the fact that the passport contains very sensitive data like the holder's name, date of birth, place of birth and so on, an attacker could use such a database for social engineering, identity theft or attacks on peoples online accounts such as Facebook as people often use these things as their passwords or security questions. Figure 4.1 shows, the difference between skimming and eavesdropping.

The biometric data stored on the passport could also help an attacker to circumvent automated border gates, where the passport check is not carried out by humans but by an automated process including facial recognition and fingerprint matching. If an attacker was able to read read a passport from the distance, he could extract the stored biometric features and use these extracted templates for attacking the automated passport checks. They could also help to circumvent automated face recognition or fingerprint matching in other environments, such as access control to private buildings.

In [46], the theoretical distance limits to eavesdropping attacks were researched. The result was that for business environments a distance of about 2 meters is the limit with an acceptable

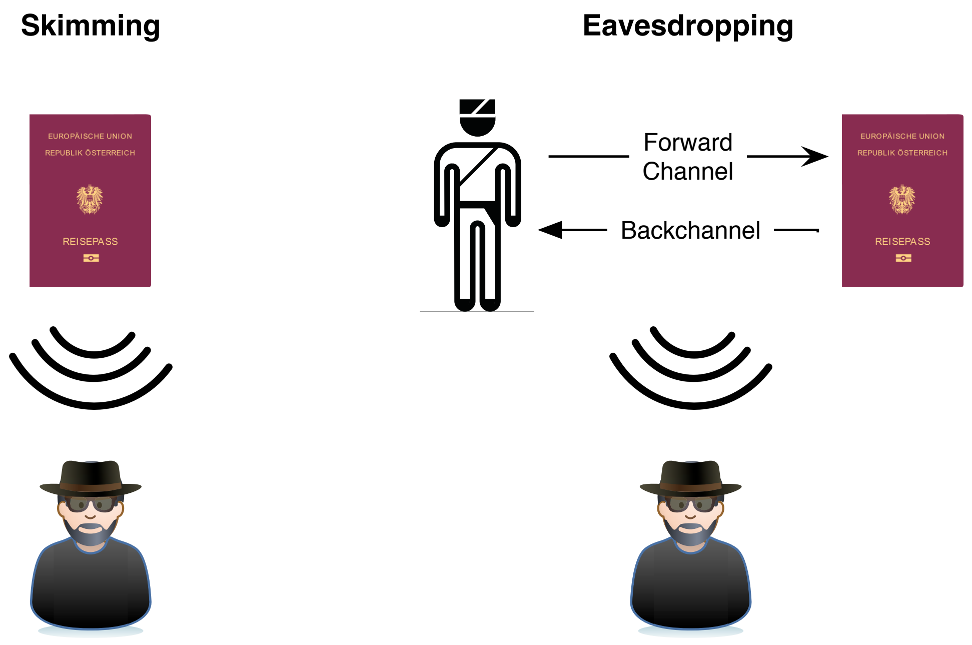


Figure 4.1: Skimming versus eavesdropping

error rate and about 7 meters for an optimal environment. The paper also states that earlier statements with distances up to 40 meters were too high either because of a too high error rate or unrealistic environment assumptions.

In [47], the German BSI achieved similar results by conducting an experiment. Therefore, they used an NXP Pegoda MF RD700 reader with a transmission power of 320 mW. For the eavesdropping attack they achieved a distance of 2.30 m with a 100% success rate. They also found, that greater distances can be achieved, if only the forward channel (see Figure 4.1) is needed. As we can see in Section 5.3 this is already sufficient to attack the Basic Access Control.

To prevent skimming and eavesdropping, the Austrian ePassport implements the Basic Access Control to ensure that the inspection system has to be able to read the passport optically before accessing the chip. For the encryption and integrity assurance of the transport channel, as described in Section 3.1, Secure Messaging is used. Extended Access Control, as described in Section 3.4, is used for a better protection of the additional biometric features such as fingerprints. In the Extended Access Control, the Terminal Authentication protocol is used to prevent skimming. The Chip Authentication protocol enables the use of a more secure key exchange for the Secure Messaging protocol.

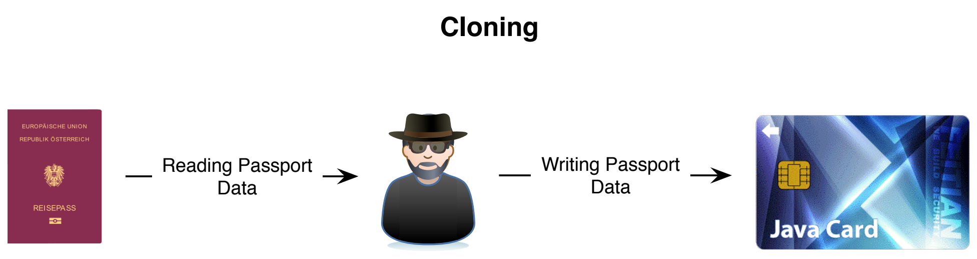


Figure 4.2: Principle of a cloning attack

4.2 Cloning

Cloning of a chip is understood as a one-to-one copy of the stored data. A cloned chip can be especially dangerous in automated border gates because they often strongly rely on the data of the chip. An attacker can only use a cloned chip, if he is able to circumvent the biometric checks like facial recognition or fingerprint matching. In earlier days, a copied or a stolen passport was used for so called look-a-likes, where an attacker used the passport of a person which looked roughly the same and it would be hard for a person to recognize the difference between the holder of the passport and the photo in the passport. As shown in [48] and many more, there is a cross-race recognition problem, which means that people are not very good at recognizing faces from different races. Look-a-like attacks may not seem reasonable for a single attacker, because it is hard to find similar looking persons for exactly one attacker. A group of attackers on the other hand could collect and clone multiple passports. The chance, that one of the attackers looks like the facial image of one of the cloned passports is much higher. Facial recognition systems as used in automated border gates on the other side have other problems, like liveness detection or the differentiation between a photograph or video and a real human face. Figure 4.2 shows the principles of a cloning attack.

The earlier Austrian ePassports without Extended Access Control (2006 - 2009) implemented Active Authentication to protect the chip against cloning. Since 2009, the Extended Access Control, especially the Chip Authentication, enables an inspection system to recognize a cloned chip. Both protocols work under the premise, that an attacker can copy all data from the chip, that is readable by an inspection system, but he is not able to copy the private key in the secure storage on the chip, where only the chip has access to. If an attacker was able, to access the private key, or the used cryptographic key was broken, a cloned passport could not be detected. Active Authentication and Chip Authentication, can not keep an attacker from copying the data from the chip, to another, but it makes it possible for an inspection system, to detect a cloned chip.

4.3 Forgery

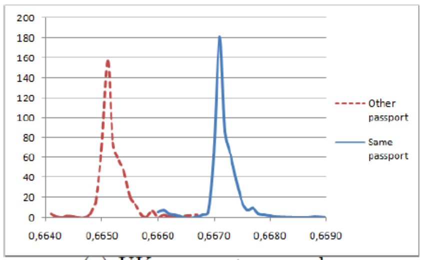
One of the most dangerous attacks on an ePassport is forgery. This means that an attacker can create an ePassport chip with arbitrary data or that he can copy an existing passport and change the data without an inspection system being able to recognize the forged data. This would completely defeat the purpose of a passport which is to reliably identify the holder of the passport by its attributes like name and date of birth. If forgery of a passport was possible, an attacker could not only cross borders under a different name, which would render border crossing procedures useless, he could also use the forged passport for different kinds of identity theft. The attacker could open up bank accounts or sign contracts under a different name.

Due to the fact that the ICAO standard [5] is open to the public a chip containing arbitrary passport data can easily be generated as shown in Section 5.7. The Passive Authentication mechanism (see Section 3.2) enables an inspection system to recognize a forged passport. To create a new passport with forged data without an inspection system recognizing, an attacker would need to be able to generate a valid EF.SOD file for the new data. Therefore, he would need to generate a Document Signer key and sign the hash values of the new data groups. Then he would need to sign this Document Signer Certificate with a valid Country Signer key. Therefore, he would either need access to a valid Country Signer key, for example by stealing one, or he would need to be able to convince the inspection system, that the used CSCA is valid, for example by somehow introducing the new CSCA into the ICAO PKD or into the German masterlist. Besides the Passive Authentication, some countries, like Austria, also have central registers, where all valid identity documents are stored and the police can look up the validity of a passport. This register is called 'Identitätsdokumentenregister (IDR)' [49]. If the attacker was able circumvent Passive Authentication, the Austrian police would still be able to detect the forged passport, because it would not be available in this central register.

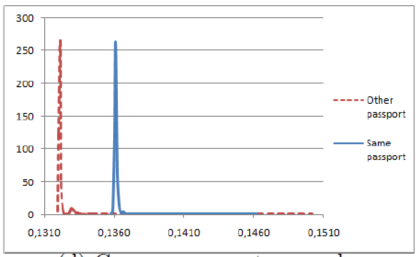
Another method an attacker could try to use to forge a passport, would be, to find a collision in the used hashing algorithms. If he could manage to create a data group with different values, which result in the same hash as the original values, the signature of the EF.SOD could stay the same and the fake passport won't be detectable by an inspection system.

4.4 Tracking

An attacker could also use a person's ePassport to track its movements. Therefore, he only needs to extract some unique attribute, which distinguishes the tracked passport from others. If the passport does not use Basic Access Control, tracking is easy because the attacker could simply read the DG1 of the tracked passport which should be globally unique. To track the passport he only has to place RFID readers at points of interest like entry and exit doors and store the captured data. Based on the captured data, a movement profile con be created. This is especially useful in places, where many people tend to carry their passport with them, like airports for example. If more documents, like driver's licenses will be equipped with RFID chips, traceability will become an even more dangerous attack vector. Besides only tracking people for



(a) UK passport on reader



(d) German passport on reader

Figure 4.3: Timing differences in British and German passports

surveillance purposes, for example, the data could also be used for marketing and advertising purposes. In [9], an RFID enabled bomb is suggested which detonates as soon as the tracked passport is near. This could enable new types of terrorist attacks, because the bomb would be inactive until the target of the attack is in the vicinity of the trigger.

In Austrian ePassports, there are multiple features that should prevent tracking attacks. The Basic Access Control prevents unauthorized access to the passport's data. As stated before, without Basic Access Control, an attacker only needs to read the DG1 to recognize the tracked passport. Another possibility to track a passport is the Unique ID (UID). According to ISO 14443 [23] each chip needs to issue a UID to execute the anti collision protocol (see Section 2.4). This feature makes it even easier to track a passport, as the attacker doesn't even have to read a data group, but only needs to contact the chip once to receive the UID. The ISO 14443 standard also specifies so called random UIDs. These UIDs start with 0x08 to indicate, that the UID is random. All of the Austrian ePassports use random UIDs to prevent tracking. Other countries used fixed UIDs in their first generation ePassports [15].

In [15], multiple methods are proposed to fingerprint passports from different countries. The first method is the Answer to Reset (ATR) value, which is used by the chip, to specify some communication parameters. Under [50], an ATR parsing service can be found, that identifies RFID products from their ATR. The service is based on a list of well known ATRs. Also the Austrien ePassport can be found in this list, and therefore identified based on its ATR. While researching the French ePassport implementation, they found another method to track a single ePassport. During the second step of the Basic Access Control, when the reader sends the *Mutual Authenticate* command, the passport answers with different error codes. If the MAC of the sent message is wrong, the chip answers with 0x6300. If the MAC is correct, but the message does not contain the correct values, the error code is 0x6A80. An attacker can make use of this information, by eavesdropping a correct *Mutual Authentication* command. If he replays the message to another French passport, he will receive 0x6300 as error code, because the MAC of the message is wrong. If he replays the message to the same passport again, he will receive 0x6A80 as error code, because, although the message contains the wrong challenge, the MAC was calculated correctly. Our tests showed, that the Austrian passport is not susceptible to this kind of attack. Despite the fact, that this behavior was only detected in the French implementation, they also found, that the passports of the other countries, although they answered always with the same error code, needed longer to respond, when the message was replayed to the same passport, than when it was replayed to another passport. Figure 4.3 shows the timing differences in British and German passports.

CHAPTER 5

Attacks

This chapter describes in detail possible attacks on the Austrian ePassport. We also implemented proofs of concept for attacks that may be practical. In this chapter, we also describe the implementation of a relay attack using two NFC-enabled smartphones and an ePassport simulator also implemented on an NFC-enabled phone.

5.1 Bruteforce Attack on the BAC

This section goes into the details of a bruteforce attack on the Basic Access Control keys also known as exhaustive search attack. This attack tries to find the correct keys by trying all different possible input parameters. In the first part, we show why the Basic Access Control is vulnerable. The later parts show different approaches to this attack, some of which are more feasible in a practical environment while some are less practicable.

Weakness of the BAC

We already discussed the key generation for the Basic Access Control in Section 3.1. The calculated keys are used for a mutual authentication between the chip and the inspection system using the block cipher 3DES. To attack the 3DES keys directly, which would mean to try to authenticate using all different 3DES keys, would take about 2^{112} authentication attempts. This seems not practical especially with limited resources. In [51], it is shown, that a known plaintext attack needs about 2^{120-t} encryptions and 2^t known plaintext. Due to the fact that plaintext-ciphertext pairs can only be collected by eavesdropping on a successful authentication session, as described later in Section 5.3, this attack is not feasible in a practical environment.

The used 3DES keys are generated with a key derivation function, which uses the SHA-1 hash value of a concatenation of the document number, date of birth and date of expiry of the passport and their check digits. The document number in Austrian ePassports consists of a leading alphanumerical character which is ’P’ at the moment. This leading character may change in the future to another one. The remaining document number consists of seven numerical characters. Both of the dates are in the format 'yymmdd'. The date of birth can be limited to 100 years because there are only two digits for the year. The date of expiry can be limited to ten years because of a ten year validity period of Austrian ePassports. The check digits do not add any entropy to the key space because they can be easily calculated with the algorithm described in [5]. Table 5.1 shows the entropy of the input parameters.

Document Number	**Date of Birth**	**Date of Expiry**
10^7	$(365 \times 100) = 36500$	$(365 \times 10) = 3650$

Table 5.1: Entropy of the input parameters of the Basic Access Control key derivation

The overall entropy for the key space is therefore $(10^7 \times 36500 \times 3650) \approx 1,3 \times 10^{15}$ which approximates to about 2^{50}. In Section 5.3 we show that this entropy can be lowered. In [31], the entropy is also calculated with approximately 56-Bit with a numeric document numbering scheme like it is the case in Austrian ePassports. In [9], the entropy for a Dutch passport was claimed to be as low as 35-Bits. Due to the fact that the referenced source is not available any more, we do not know the assumptions and could therefore not verify this.

In our analysis, we found out that the document number of the Austrian ePassport is in a sequential order. Therefore, it depends on the date of expiry and we can put these two parameters into a relation. One possibility to lower the entropy with this information would be to compile a database with at least one passport for every day since the beginning of the ePassport. In the annual report of the Austrian State Printing House, where the Austrian ePassports are printed we can find, that in the business year 2013 they printed 1.2 Million ePassports. This means that there are about 3.300 passports printed every day. We can now say that all document numbers for a given date of expiry can be calculated by looking up the date in the precomputed list. Most of the document numbers will be between this value minus 3300 and this value plus 3300 ($List[DoE] - 3300 \leq documentnumber \leq List[DoE] + 3300$). As the compilation of such a list takes a long time, an attacker can take advantage of the fact, that the correlation between the date of expiry and the document number is almost linear and calculate such a list by interpolating all the document numbers from a collection of valid passports. The more passports an attacker is able to collect, the better such an interpolated list will be.

5.2 Online Bruteforce Attack

The first approach to attack Basic Access Control using bruteforce is to try an online bruteforce attack, which means that all possible keys are used to try a full authentication with the attacked passport. We built a script that calculates a key candidate first (K_{enc} and K_{mac}) and then sends the *Get Challenge* command to the passport and then builds E_{IFD} and M_{IFD}. Using these values it sends the *Mutual Authenticate* command to the chip. If the answer of the chip has its status word set to 0x9000 the key candidate is actually the right key. If the status word is

```
input  : A list of possible key generation parameters
output: The correct key
rndifd ← generateRandom(8);
kifd ← generateRandom(16);
for $i \leftarrow 0$ to $i < input \rightarrow length$ do
    (kenc, kmac) ← caluclateKeys(input[i]);
    rndicc ← sendGetRandom();
    s ← rndicc + rndifd + kifd;
    eifd ← encrypt(s);
    mifd ← calculateMac(eifd);
    if sendMutualAuthenticate(eifd + mifd) $== 0x9000$ then
        return $input[i]$ ;
    end
end
```

Algorithm 5.1: Online bruteforce attack in pseudocode

different, the next key candidate is calculated. The idea of this approach is shown in Algorithm 5.1. For a better understanding, we assumed that all the possible key generation parameters are precomputed in a list.

We timed one hundred cycles of this loop to find out about how long the full run of the script would take. If we neglect the duration of the computations on the reader side like calculation of K_{enc} and K_{mac} and encryption of S, one run would still take about 60ms. To calculate the key it would take about 2 million years. There is no way to speed up the processing speed of the chip, which is causing the low throughput. We also tried to parallelize the attack. In our first try, we started multiple threads for the authentication attempts, trying to find out if the communication could be accelerated. In fact we even lost a few milliseconds because of the threading overhead. Another attempt, to speed up the attack was, to use logical channels, as described in Section 2.4. Therefore, we used the mutlithreaded implementation and tried to open a new logical channel for every authentication attempt. This implementation was not successful, because the passport chip in our test passports did not support logical channels. Due to the fact, that we did not find any other method to parallelize the attack, the only way to speed it up is by lowering the entropy further.

Using the knowledge of the sequential document number for the attack could bring the calculation of the key to a duration of 660 years, which is a lot less than before but still not practical.

To lower the entropy even further, we could argue, that during an online bruteforce attack, the attacker is physically close to the holder of the attacked passport. This gives him the possibility to take an educated guess of the holder's age. Assuming the error of this guess can be limited to about 10 years the entropy of the date of birth would be 3650. The calculation of the key would still take about 66 years even with these assumptions.

In our opinion, an online bruteforce attack on the Basic Access control is not feasible in practical environments. Even if the further limitations of the entropy can be justified and have been used before (e.g. in [9]), the time to calculate the key is still too long as to be exploitable in a practical environment. Some attack scenarios in [12] argue that the date of birth could be obtained from other sources. If this assumption is true, the data on the passport would possibly not give an attacker much information that he cannot obtain from this other source.

5.3 Offline Bruteforce Attack

For an offline bruteforce attack, the attacked passport itself is not needed. It always depends on a recorded (eavesdropped) session between a reader and a chip. The attack then tries to calculate the Basic Access Control keys from this recorded communication. In [12], two possibilities are presented how such an attack can be executed.

Attacking K_{enc}

For this attack to succeed, an attacker has to record at least the first three messages of the Basic Access Control protocol.

- The random nonce of the chip (RND_{ICC})
- The response from the legitimate reader (E_{IFD} & M_{IFD})
- The response from the chip (E_{ICC} & M_{ICC})

With these three messages the attacker can calculate a possible candidate for K_{enc}, which we call $K*$, and use this to decrypt E_{ICC}. He can now compare the 8 most significant bytes of decrypted plaintext with RND_{ICC}. If this is no match, the attacker tries the next key candidate. If there is a match he can also decrypt E_{IFD} and compare RND_{ICC} with bytes 9 to 16 of the decrypted E_{IFD}. If they are the same, the attacker could then compare RND_{IFD} from both decrypted messages to be absolutely sure that the key candidate is the right key.

For the first part which has to be done for every possible key candidate two SHA-1 calculations are needed for the generation of $K*$. Additionally, one 3DES decryption has to be done. For the second part, which has to be done only for very strong suspects for the right key, one additional 3DES is decryption necessary. Due to the fact that we cannot predict how high the possibility is that the first comparison matches for a wrong key, we cannot predict how often the second part has to be done. In [12], this possibility is completely neglected.

Attacking K_{mac}

Another approach is necessary especially when an attacker is not able to record the backchannel (Chip $\rightarrow$ Reader). In this case, the attacker does not try to calculate the encryption key K_{enc}

but the MAC key K_{mac}. As in the first approach the attacker calculates a key candidate for K_{mac} $K*$. With this key he calculates the MAC for E_{IFD} according to [5]. He then compares the calculated MAC with the captured M_{IFD}. If there is no match, the attacker calculates the next key candidate. If there is a match, the right K_{MAC} was found. Using the key generation parameter that was used to generate the found K_{mac}, the K_{enc} can be calculated immediately.

For this alternative attack, the attacker has to do two SHA-1 calculations for the generation of $K*$. Additionally, to calculate the MAC the attacker has to calculate four DES encryptions and one 3DES encryption.

Comparison of the Attack Alternatives

We already described one drawback of the second approach which are the four additional DES decryptions. The big advantage of the second approach on the other hand is that an attacker only has to capture the forward channel and not the backchannel, which is often easier and can be done from further away, as the forward channel sends a stronger signal [46]. The information an attacker can gain from only recording the forward channel is limited because he can only calculate the key and has to use this key in combination with a skimming device to read the attacked passport's data. If the attacker has a recording of a full session, including the backchannel, he can reconstruct the session key KS_{enc} and use it to decrypt the whole session. As most of the time an inspection system reads out all the data stored on the passport, the attacker immediately has all the information read by the inspection system.

Single Threaded Implementation

To get the magnitude of how long the offline bruteforce attack would take, we did a naive implementation of the first attack using Java. We used a standard Intel Pentium i5 processor for the timing of the attack. This naive implementation uses no optimization. It just loops through all possible document numbers, dates of birth and dates of expiry. If the correct key is found it stops and prints the underlying key parameters.

One run took about $23\mu s$. If we use the entropy without the further limitations of Section 5.1 it would take approximately 372 years to calculate the key. Taking advantage of the correlation between document number and date of expiry, the attack would take about 235 days $((36500 * 3650 * 6600 * 0.000023)/(60 * 60 * 24))$.

Parallelized Implementation

Analyzing the algorithm in Section 5.3, we saw that it could be easily parallelized. There are multiple methods, how this could be done. One way to achieve parallelization of the Algorithm would be a simple multithreaded implementation. Therefore, for each attempt to decrypt the ciphertext, a new thread is created. These threads can be executed by the CPU in parallel. Using

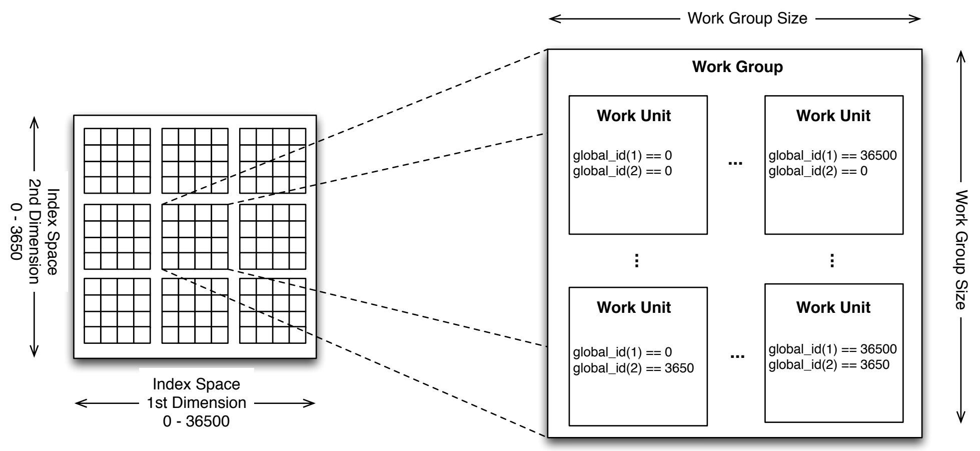


Figure 5.1: Example of an index space with work units and work groups [53]

this method, machines with multiple CPUs or multicore CPUs can be fully taken advantage of.

In previous years General Purpose Graphical Processor (GPGPU) programming has been getting more and more popular for parallel computing. As our goal was to find attacks that are practical without the need for specialized expensive hardware and most modern computers have a GPU that is capable of GPGPU programming, we decided to make use of this fact.

GPUs have been improving very fast in previous years, mainly because of the massive requirements of modern computer games. To meet these requirements, GPUs have been designed for massive multithreading. The AMD Radeon HD 6850 used in this thesis for example has 960 Stream Processors[1] (SP). Although SPs are not directly comparable to CPU cores, because of different architectures, it shows that GPUs are highly parallelizable.

To make use of the GPGPU capabilities of modern GPUs, a framework is needed to control the computations. At the moment there are two popular frameworks for GPGPU programming. One is the Compute Unified Device Architecture (CUDA) [52], developed by the GPU manufacturer NVIDIA Corp. The other framework is the Open Computing Language (OpenCL) [53]. While CUDA only works for NVIDIA GPUs, OpenCL is an open standard with implementations on various architectures like AMD GPUs, NVIDIA GPUs, Intel GPUs and Intel CPUs. Because of the open nature we chose OpenCL for developing this attack.

Introduction to OpenCL

An OpenCL program consists of two main parts. One part is the kernel which is executed on one or more devices. A device in this case can be any computing unit compatible with OpenCL. This could for example be a GPU, CPU or any other processor such as a Digital Signal Processor (DSP) or similar. The other part is the host program, which defines the context for the kernels and dispatches them to a device. When a kernel is dispatched to the device, an index space is defined. This index space can have up to 3 dimensions. For each point in this index space an instance of a kernel is executed. This instance is called a work unit. Each work unit has a global ID which marks the point in the index space for this work unit. Multiple work units are organized into work groups. Figure 5.1 shows how the index space is decomposed into work groups and work items.

OpenCL differentiates between four types of memory regions, whereas the fastest of the memory types is the private memory. The slowest memory is the global memory.

- **Global Memory** The global memory region allows all work units read and write access. Global memory can also be accessed by the host.
- **Constant Memory** The constant memory region is a special region of global memory, which remains constant and can be only read by the work units
- **Local Memory** The local memory region is shared between all work units of one work group.
- **Private Memory** The private memory region can only be access by the work unit itself and is not shared with any other work unit.

OpenCL Kernel

The first step to create the program for the attack is to find the part of the algorithm that stays the same over all iterations. Looking at the algorithm described in Section 5.3 we can see that this is the part inside the for loop which consists of the key derivation and the decryption of E_{ICC}. This part is the optimal candidate for the kernel. Algorithm 5.2 shows a simplified version of the kernel.

In the first part of the kernel, the MRZ is calculated from the input parameters document number, date of birth and date of expiry. As described later these three parameters are passed to the kernel as integers using the unique global ids of a work unit. To convert the integers to a string representation of the date of birth and date of expiry we used two lookup tables. The first lookup assigns each number between 0 and 9 its string representation. For example the first element is '01', the second element is '02' and so on. This table is used for the calculation of the

[1]`http://www.amd.com/us/products/desktop/graphics/amd-radeon-hd-6000/hd-6850/pages/amd-radeon-hd-6850-overview.aspx#2`

```
input  : Cadidate for MRZ (document number, date of birth, date of expiry), $E_{ICC}$
output: The correct key
mrz ← createMRZ(document number, date of birth, date of expiry) ;
kseed ← calculateSHA1(mrz) ;
kenc ← calculateSHA1($MSB_{16}$(kseed) & 0x00 00 00 01) ;
if rndicc == 3DESDecrypt(kenc, eicc)[8 : 16] then
    $output$ = mrz ;
end
```

Algorithm 5.2: OpenCL kernel in pseudocode

year. The second lookup table assigns a number to each day of the year. For example the first element in the array is ’0101’, the second is ’0102’ and so on. For a cleaner implementation we ignored the possibility of a leap year. To calculate the year, we divided the integer by 365 and used the floor function (which returns the next smaller integer of a decimal number). The day of the year is then obtained by calculating the remainder of the division of the passed integer by 365 using the modulo function. To calculate the date of expiry, the current date is also passed as an argument to the kernel, as the date of expiry is at most 10 years in the future of the current date. In the last step of the MRZ generation, the check digits of the individual fields are calculated. The algorithm for calculating these check digits is also specified in the ICAO 9303 [5]. First, all digits are alternatingly multiplied by 7, 3 and 1. So the first digit is multiplied by 7, the second by 3 and the third by 1 and so on. The products are then summed-up and divided modulo 10. Alphanumeric characters are represented by a numerical value. ’A’ is represented by 10, ’B’ by 11 and so on.

The next part of the OpenCL kernel is the calculation of the SHA-1 hash, which is specified in [54]. The SHA-1 hash consists of 80 rounds, whereas each round consists of multiple additions, leftrotate functions and bitwise operation. As a basis for our implementation, we took the OpenCL SHA-1 implementation of the well known Open Source bruteforce software John the Ripper[2]. This implementation is already very efficient. Due to the fact, that we exactly know the input length of the two SHA-1 operations, we could further optimize the calculation. This optimization could be achieved, because some of the calculations depend on the length of the input to the SHA-1 function. Therefore, loops with dynamic iterations were used. Because the length of the input is known beforehand, we replaced these loops with fixed length loops. There is also a shortcut for calculating a SHA-1, which can be used, if the input is less the 20 bytes long. We could make use of this shortcut for one hash calculation.

The last part of the kernel is the decryption of E_{IFD}. As mode of operation a two-key 3DES in CBC mode with a zero initialization vector, as specified in the ICAO 9303, is used. As basis for this implementation we used the 3DES algorithm from the PolarSSL[3] library. This is already a very efficient implementation. To optimize the algorithm further we tried to unroll as many

[2]http://www.openwall.com/john/
[3]https://polarssl.org/

loops as possible, because loops, especially without fixed iterations, are not very efficient inside OpenCL kernels.

Host Application

To call the OpenCL kernel, we needed to create the host application. This application runs on the CPU of the host machine and is responsible for managing the environment for the OpenCL kernel. In our case, we used C++ to implement the application, but there are OpenCL libraries for many other programming languages. In the first step the host application looks for a compatible device, to execute the kernel on. As described earlier, OpenCL kernels can not only run on GPUs, but can also be used to better utilize multicore CPUs or other processors like DSPs. After creating an object to communicate with the found device, multiple buffers have to be created. Normally buffers for input parameters would have to be created as well on the host machine, as well as on the OpenCL device. Due to the fact, that the host application cannot access the device's memory directly, the buffers have to be created first in the memory of the host application and then copied to the OpenCL device using the special function *clCreateBuffer(...)*. In our case, the kernel does not take any input parameters, so we did not have to create any input buffers. The same procedure is done for the output parameters. If the kernel finds the correct key, it puts the MRZ of the current key candidate into an output buffer. To access this buffer, the host application has to copy the output buffer back from the device's memory to the host's memory. Due to the fact, that all running kernel instances need to be able to access these parameters, they have to be put into the global memory of the device, which is the slowest memory in an OpenCL device. Therefore, the access to these buffers should be kept to a minimum. If the values have to be accessed very often during the kernel's calculation, it should be considered, if it may be faster to copy the input parameter into the device's local memory.

The next step is to define the index space. We decided to use a three dimensional index space with 36500 x 3650 x 100000000 elements, where the first dimension stands for the date of birth (which is 100×365), the second dimension represents the date of expiry (10×365) and the third dimension the document number. In Figure 5.1 the first two dimensions of the index space can be seen. For each of the elements in the index space a work unit is created. Each work unit is an instance of the kernel. Due to the fact that we defined our index space to be exactly the size of the key space that has to be searched, each work unit represents exactly one try in the bruteforce attack. Each work unit can get its index from the *get_global_id(int)* function with the dimension as its parameter. This three dimensional index can then be converted to the three input parameters for the key generation as described Section 5.3.

The last step of the host application is to compile the kernel and prepare it for execution. Therefore, a work group size has to be defined. A work group is a group of work units which are executed on the same streaming processor inside the GPU. The work group size is very critical to the overall speed of the calculation. There is also a way to let the compiler choose the optimal work group size. We tried different work group sizes, according to best practice guides and tutorials, but we could not optimize the automatically chosen values any further. The host application then queues the kernel execution with the chosen parameters and starts the queue.

The application waits for the queue to finish and copies back the output buffer to the host memory. The queue could also be run asynchronously with the host application doing further work. This makes it possible to split up the index space and build multiple queues, which could then be executed on multiple GPUs on one host.

Performance Analysis

With our program we were able to calculate $5,5 \times 10^6$ or 2^{22} keys per second on our test machine using a three year old AMD Radeon HD 6850. This is about twice as much keys per second as was achieved in [12] with one core of the COPACOBANA[4] hardware codebreaker. To search the whole key space with this performance would still take seven years. Taking into account the limitations to the document numbers proposed in Section 5.1 it would take a maximum of 44 hours to calculate the correct key. To speed up the bruteforce attack, multiple GPUs can be used in parallel. Depending on the hardware, the performance in this case will increase linearly.

One possibility to speed up the computation, would be to distribute the workload over multiple hosts, with potentially multiple GPUs. To do this, the Berkeley Open Infrastructure for Network Computing (BOINC) could be used, where a big group of people volunteer the idle time of their computers, to participate in different projects. To make use of this network, a project would need to be set up. This project can either be a private project or a public project. Well known public projects are for example the SETI@Home project with more than 1.4 million participating users. To calculate the Basic Access Control keys, without any further limitations of the entropy, within 10 minutes, our implementation would need around 350,000 participating hosts. If all users of the SETI@Home project[5] would participate to calculate one Basic Access Control key, the calculation would take about three minutes. Figure 5.2 should visualize these proportions.

In [55], they profiled a botnet and also found prices for different amount of bots. They found that a 'world mix' package, including hosts from undefined locations, costs 25 $ per 1,000 bots and 200 $ per 10,000 bots. This means that if we were to buy a botnet to execute the described attack on the Basic Access Control, it would cost 1.6 $ to achieve the same performance as in [12], which would take about 64 hosts. To buy the 350,000 hosts needed to calculate the key in within 10 minutes, it would cost 7,000 $.

Key Precomputation

During the implementation of the offline bruteforce attack, we saw, that a large part, that is the calculation of the key, is done for every attack and is always the same. This part consists of the calculation of the check digits in the MRZ and two consecutive SHA-1 operations. Especially the SHA-1 operations take a lot of time, as every SHA-1 calculation consists of 80 rounds. To

[4]`http://www.copacobana.org/`

[5]`http://boincstats.com/en/stats/0/project/detail`

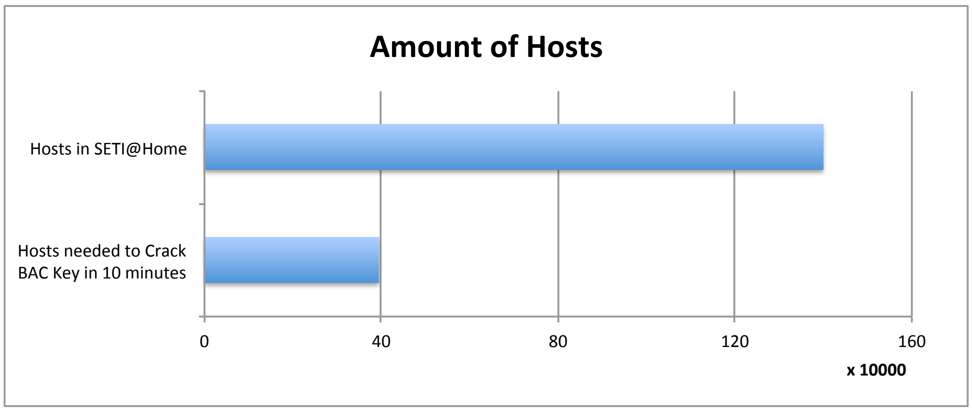


Figure 5.2: The amount of hosts needed to crack a BAC key in 10 minutes versus the amount of hosts in the SETI@Home project

speed up the bruteforce attack, all possible keys could be precomputed and stored in a file. During the bruteforce attack, the already precomputed keys are used and only one 3DES decryption and one comparison has to be done. This is called a space-time tradeoff as the attack uses more storage space, but takes less time to complete.

To see if it is feasible to precompute all the possible Basic Access Control keys, we calculated the amount of keys. Without any limitations, the key space is, as already discussed, approximately $1,3 \times 10^{15}$. Each key represents the 16 most significant bytes of a SHA-1 hash, so each key takes up 16 bytes in space, which adds up to about $1,9 \times 10^{7}$ Terabyte. To limit the amount of space used by the precomputed keys, we can make use of the correlation between document number and date of expiry. If we take this into account we can limit the space needed for the precomputed keys to 13 Terabyte. In this case the precomputation of the keys is already feasible, provided the keys can be read out faster from the storage medium than calculated on-the-fly.

5.4 Relay Attack

In the previous years more and more automated border gates were installed in airports around the world. There are installations for example in Frankfurt, London, Amsterdam and Paris [56]. At these gates ePassports are used to enter the gate and then a face recognition system matches the travelers face to the face on the passport. With this unattended border crossing, relay attacks, as described for many other RFID systems [57], become a problem. A relay attack is an attack, where the genuine passport is at a different location than the inspection system. The attacker uses a specialized chip which is used as a proxy. This proxy sends all commands from the genuine reader to another reader under the control of the attacker, but at the location of the genuine passport. The attacker's reader, also called a mole, then sends all answers from the genuine

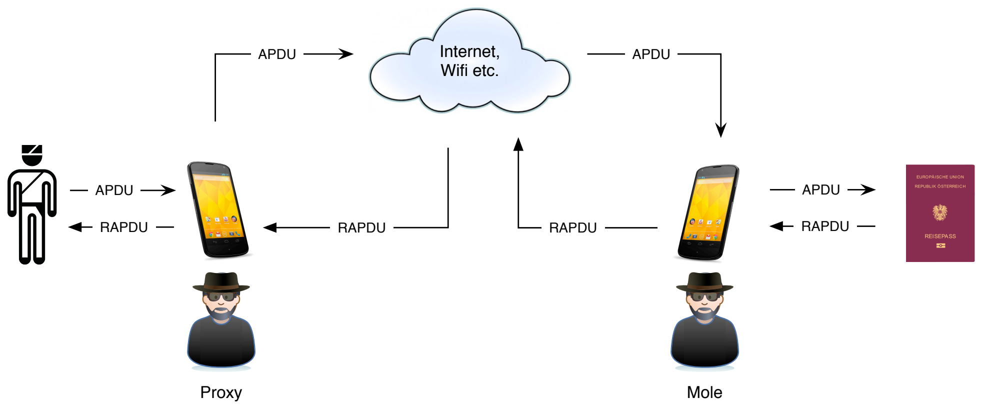


Figure 5.3: Basic principle of a relay attack

passport back to the proxy and the proxy sends them back to the genuine reader. This attack is also known as the Grandmaster Chess Attack [10]. Figure 5.3 shows the basic principle of a relay attack.

To be able to execute such a relay attack, a special hardware is needed to act as a proxy. This proxy has to be able to emulate an RFID tag, in this case an ePassport, and forward all the commands to the mole. Commercial products which are able to do this, are very often part of some RFID testing equipment and are therefore rather expensive. There has been some open source hardware around like the OpenPICC[6]. The disadvantage of this solution is, that it cannot be used as a standalone device, as there has to be a host device like a notebook to control it.

As already discussed in [57], it is hard for an RFID system to prevent such attacks, because the attacker has access to the genuine passport. Some mechanisms in the ISO 14443 standard even enable this kind of attacks, because the tag can always tell the inspection system how long it should wait for an answer using the so called Waiting Time eXtension (WTX) command. Another way to stall the inspection system is to set a very high Frame Waiting Time (FWT) during the anticollision protocol.

An inspection system could enforce a shorter FWT to give the passport less time to answer, but due to the fact that this countermeasure would make the inspection system incompatible to the ISO 14443 standard, it would be impractical.

In [57], two ways to prevent relay attacks are proposed. One way to make these kinds of attacks harder would be to cryptographically secure application parameters like the FWT which defines the time the inspection systems waits for the next frame to receive. This way the relay proxy cannot stall the inspection system by changing these application parameters if the connection between the mole and the proxy is too slow to answer in the correct time as it does not

[6]`http://www.openpcd.org/OpenPICC_RFID_Emulator_Project`

know the cryptographic keys used to secure these application parameters. A problem with this solution is, that not all ways to stall the inspection system are in these application parameters. The WTX command for example can still be sent by the proxy. Another disadvantage of this method is, that in order to implement it, the ISO 14443 standard would have to be changed to support this kind of security. Other methods shown are distance bounding protocols as for example proposed in [58]. These protocols make use of application level mechanisms to make it possible for an inspection system to verify the distance between the reader and the passport. This could be achieved for example if the passport chip was equipped with a Global Positioning System (GPS) receiver.

5.5 Index Manipulation Attack

This attack was published in 2008 at the Black Hat conference in Tokyo, Japan [13]. It shows that if an inspection system is implemented strictly according to the ICAO 9303, first the EF.COM is read. The EF.COM includes a list of tags that represent the available files on the passport. If this tag list includes the tag $'6F'$ which means that the DG15 for the Active Authentication public key is available, the Active Authentication is performed. If it is not included the Active Authentication is not performed. Table 5.2 shows the file index in the EF.COM file.

Length	**DG1**	**DG2**	**DG11**	**DG12**	**DG15**
05	61	75	6B	6C	6F

Table 5.2: File index in the EF.COM file

The problem with this implementation is, that the EF.COM is not protected by the Passive Authentication and can therefore be altered without the inspection system noticing. In the published attack, the EF.COM was altered and the tag for the Active Authentication public key was removed. Therefore, the inspection system would not perform the Active Authentication because the EF.COM does not indicate that the passport supports it. If an inspection system is vulnerable to this kind of attack, it would work with the Chip Authentication mechanism as well. In Table 5.2 for example, only the tag ’6F’ has to be removed and the length field has to be altered to ’04’ instead of ’05’.

A very simple solution for this problem would be not to check the available files in the unprotected EF.COM but in the EF.SOD. If the EF.SOD file would be altered, it can be recognized, when performing the Passive Authentication because the signature could not be verified anymore. In our tests we found that at least the Golden Reader Platinum Edition, which is often used as a reference implementation, still uses this insecure detection mechanism per default. We also found that there is the possibility to change these default settings so, that the EF.SOD is used as file index. Using this setting makes the reader to ignore the EF.COM and read the EF.SOD as first file and verify the signature. If the hash of a data group is not in the EF.SOD, the file really does not exist, because if an attacker would have just deleted the hash from the EF.SOD file, the signature verification would have failed.

5.6 Attack by Once Valid Readers

In [59], it was shown that the lack of an accurate internal clock can be exploited by an attacker. The internal clock of an ePassport is only updated if it is presented with a valid terminal certificate during Terminal Authentication if the issuance date of the certificate is after the date of the internal clock in the ePassport. This can be a problem especially for infrequent travelers, because the date is not getting updated very often. If an attacker manages to steal a DV or a terminal certificate, this certificate can be used on these passports even if the certificate is already expired. This might be especially useful for an attacker if the stolen certificates have a very short validity period.

In [11], a German security researcher showed another way to exploit the lack of an accurate internal clock and the described update procedure. An attacker who managed to steal a valid CV or terminal certificate and create a CV or terminal certificate with an issuance date far in the future could set the internal clock to this future date. The passport won't accept any further CV or terminal certificates, because in the ePassport's time they have already expired.

Both of these attacks require for an attacker to get in the possession of a valid terminal or CV certificate. Even if this is possible, it is not very likely.

5.7 Insecure Inspection Systems

In 2008, there have been news about the Dutch hacker group The Hackers Choice (THC), who claimed to have hacked the electronic passports. They said that they were able to pass a passport check as Elvis Presley. They also published a video showing them how they put the passport on the reader and it shows Elvis Presley's name and photo on the display without raising any alarms. In [60], it is shown why this attack worked. The problem was that the inspection system used, which was a self check-in terminal in this case, did not check the the validity of the CSCA certificate. If the CSCA certificate is not checked by the inspection system, an attacker could simply create his own CSCA. Using this CSCA he could use the publicly available ICAO 9303 to create an ePassport with arbitrary data and a chosen photo. In this case, they chose the data of Elvis Presley. Although this attack was only possible due to a bug in the inspection system, the checking of the CSCA is not a trivial task. The problem is, that at the moment there are only 37 countries, which are part of the ICAO's public key directory, but there are 100 countries issuing ePassports [36]. CSCA certificates, which are not in the public key directory have to be exchanged in a bilateral way. An attacker could easily create a new CSCA using a country code for a country that is not part of the public key directory and the country he wants to travel to is not likely to have bilaterally exchanged their certificates. It then depends on the implementation and the purpose of the inspection system, if these certificates are accepted.

5.8 Android Implementation

One of the goals of this thesis was to analyze if it is feasible to implement some of the attacks found in this section on a Near Field Communication (NFC) enabled Android Smartphone. NFC is a technology based on RFID and is specified in ISO 18092/ECMA 340[61] and ISO/ECMA 352 [61]. NFC is compatible with existing RFID standards like ISO 14443 [23] and others. There are three modes of operation defined in the NFC standards.

- **Reader / Writer Mode**
  In Reader / Writer mode the NFC device (for example a smartphone) acts like a standard RFID reader. It can send and receive APDUs to and from an NFC tag.
- **Card Emulation Mode**
  In Card Emulation mode the NFC device acts like an NFC tag and can receive APDUs from an RFID reader and respond to them.
- **Peer to Peer Mode**
  In Peer to Peer mode two NFC devices can communicate with each other. In this case both devices act alternating as reader or as a tag.

To show the feasibility, we created an Android application with mainly two functions. It is able to perform a relay attack and it can be used to simulate a passport that has been read before either using the application itself, or using another software and copying the data to the smartphone.

Relay Attack

As already described in Section 5.4, a relay attack consists of two parts, the mole and the proxy. To implement the mole, we used the Reader / Writer mode capability of our Android Smartphone. This mode has been implemented in the Android NFC stack since Version 2.3.3 [62]. The mole waits for a passport to be placed in the RF field of the NFC chip. After this has happened, it immediately proceeds with the anticollision and selection procedure, as this cannot be controlled from within the application. It then opens a socket and waits for a connection from the proxy.

For the application to be notified if there is a passport in the RF field, Android uses a tag dispatch system. The application registers for a specific tag technology and gets called if a tag, using this technology, is detected. Such technologies for example can be Mifare Classic or ISO 14443 compatible tags. In our case the application registers for the ISO Dep technology, which represents an ISO 14443 compatible tag. After the application is notified it can exchange APDUs with the tag. [62]

To implement the proxy, the phone has to be set into the Card Emulation mode. Until recently Card Emulation mode in Android could only be used with a Secure Element (SE). A Secure Element itself is a special type of smartcard, which can be used to store cryptographic keys for example. Most Secure Elements make use of the JavaCard technology, which can be programmed using the Java programming language. There are mainly three possible types of Secure Elements at the moment.

- **Embedded Secure Element**
  In this case the Secure Element is most often implemented as a dedicated smartcard token integrated circuit (IC) inside the phone.

- **Subscriber Identity Application module**
  Here the Secure Element is integrated into the SIM Card of the mobile phone. This way is the preferred way for mobile phone operators, because they would have control over the applications running on the Secure Element. Due to the fact, that there is still a high percentage of mobile phones, which do not support NFC, there have been developments, that not only the Secure Element is put on the SIM Card, but also the NFC antenna and controller are put directly on the SIM Card. This would make NFC a completely manufacturer independent technology.

- **Removable Memory Component**
  This type of Secure Element is implemented on a removable memory component like a Secure Digital Card (SD). Due to the fact, that less and less smartphones are equipped with SD Card slots, this type is very seldom used.

In these older versions of Android, if the smartphone is set into card emulation mode, the NFC chip is directly connected to the Secure Element. The external card reader in fact talks directly to the application on the Secure Element and the NFC chip only relays this communication. The problem for implementing the proxy with these Android versions, is, that the application itself never receives the APDUs from the external card reader. One way to implement the proxy in spite of this fact, would be to make use of the Secure Element. The code could be implemented directly on the Secure Element using JavaCard for example. This implementation would then relay the APDUs from the card reader to the application running on the phone. This application would then redirect them, for example over the Internet, to the mole. Because of the, at the moment, few commercial NFC applications, no process has been agreed on, how an application can get access to the Secure Element. Therefore, we were not able to test this method of implementing the proxy.

In 2012, a programmer of the very popular open alternative to the official Android build CyanogenMod[7], found, that in the source code of the Android operating system a card emulation mode, which can be used from an Android application, was already implemented, but commented out. After activating the necessary lines and some minor modifications, the card emulation mode could be used. The disadvantage of this way is, that an Android user who wants to use the application has to have a rooted phone with CyanogenMod installed.

[7]`http://www.cyanogenmod.org`

```
public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {
  if (mode == MODE_SIMULATOR) {
    ...

  } else {
    new ProcessAPDUTask().execute(commandApdu);
    return null;
  }
}
```

Listing 5.1: processCommandApdu(...) method of the HostAPDUservice

With Android 4.4, Google changed its way regarding NFC by officially announcing the Host Card Emulation (HCE) mode. HCE enables applications to register Application Identifiers (AIDs) with the operating system in the manifest file. If an external card reader selects this AID, this application is called. This enables developers to port smartcard systems to a mobile phone, because no access to a Secure Element is needed anymore. This mode also allows to do exactly what is needed for the implementation of the relay attack. For the implementation of the relay proxy we had to build an Android service, which extends the HostAPDUService class. In this service we had to override three methods. On creation of the the service, the *onCreate(...)* is called. In this method a socket is created, which is used to connect to the mole. The most important method is the *processCommandApdu(...)*. In this method the service receives the APDU sent by the external card reader and using the socket created before it relays all APDUs to the mole. It then waits for the moles response and returns it back to the external card reader.

On creation of the HostAPDUService, it is fist checked, if the service runs in simulator mode or in relay mode. The simulator mode is used for the Passport Simulator described in more detail in the next section. In relay mode it checks if WiFi Direct is available. If it is, the connection to the mole is established. The *processCommandApdu(...)* method (see Listing 5.1) takes the current APDU and relays it to the ProcessAPDUTask.

The ProcessAPDUTask, responsible for procession of the APDU itself, is realized as a AsyncTask. This task is running in it's own thread. The doInBackground(...) method represents the run(...) method of a thread. For the communication between the mole and the proxy, a simple transmission protocol is used. The first two bytes of a message denotes the length of the data that is following. Listing 5.2 shows the implementation of the ProcessAPDUTask.

As described before, the mole waits for the NFC dispatch. When the *onTagDiscovered(...)* callback is called, the mole starts again an AsyncTask (ServerSocketTask) that is responsible for communicating with the proxy. On our first tests, we experienced connection losses when communicating with the passport. We found that there is a bug in the NFC stack developed for the Broadcom BCM20793 NFC controller used in the Nexus 4 and the BCM20794 NFC controller used in the Galaxy S4. To check if the tag, the controller is communicating with, is still present, it uses a so called presence check. In the ISO14443 different methods for such a presence check

```
private class ProcessAPDUTask extends AsyncTask<byte[], Integer, byte[]> {
  protected byte[] doInBackground(byte[]... params) {
    byte[] commandApdu = params[0];
    try {
      // First write the length of the APDU

      byte[] commandLength = Util
          .shortToByteArray((short) commandApdu.length);
      socketOutputStream.write(commandLength);
      // and then the rest
      socketOutputStream.write(commandApdu, 0, commandApdu.length);

      // First 2 bytes to arrive is the length
      short responseLength;
      byte[] readBuffer = new byte[2];
      socketInputStream.read(readBuffer);
      responseLength = Util.byteArrayToShort(readBuffer);
      if (responseLength < 0)
        throw new IOException("Socket closed on the other side.");
      readBuffer = new byte[responseLength];
      // now read the rest of the response
      socketInputStream.read(readBuffer, 0, responseLength);
      sendResponseApdu(readBuffer);
      return readBuffer;
    } catch (IOException e) {
      Logger.getLogger("MRTDProxy").info(
          "IOException thrown. " + e.getMessage());
      return new byte[] { (byte) 0x6F, 0x00 };
    }
  }
}
```

Listing 5.2: ProcessAPDUTask of the HostAPDUService in relay mode

are defined. One method would be for the controller to send an empty I-Block to the chip. The chip should answer with an empty I-Block itself. The Broadcom NFC stack does not adhere to the specification in this case, but sends a *Read Binary* command to the passport. Due to the fact, that there is already a Secure Messaging channel opened with the passport and the *Read Binary* is unencrypted, the passport closes the Secure Messaging channel. When the mole then sends the next command it receives from the proxy to the passport, the command is rejected, because the used Secure Messaging channel is already closed. This bug has already been reported to Google. Although there is no fix at the time of writing, we found a workaround. Since Android 4.4 there is a method *enableReaderMode(...)*. This reader mode makes it possible to do a more detailed configuration of the NFC chip. One of the options of the reader mode is, to set a higher timeout for this presence check. If set to a value higher than 5,000 the presence check does not influence the mole. Listing 5.3

To test our implementation, we used a Google Nexus 4 running Android 4.4.2 as the proxy

```java
protected void onResume() {
  super.onResume();

  Bundle options = new Bundle();
  options.putInt(NfcAdapter.EXTRA_READER_PRESENCE_CHECK_DELAY, 10000);

  mAdapter.enableReaderMode(this, this, NfcAdapter.FLAG_READER_NFC_A
      | NfcAdapter.FLAG_READER_SKIP_NDEF_CHECK, options);
}
```

Listing 5.3: Workaround for the presence check bug in the Broadcom NFC stack

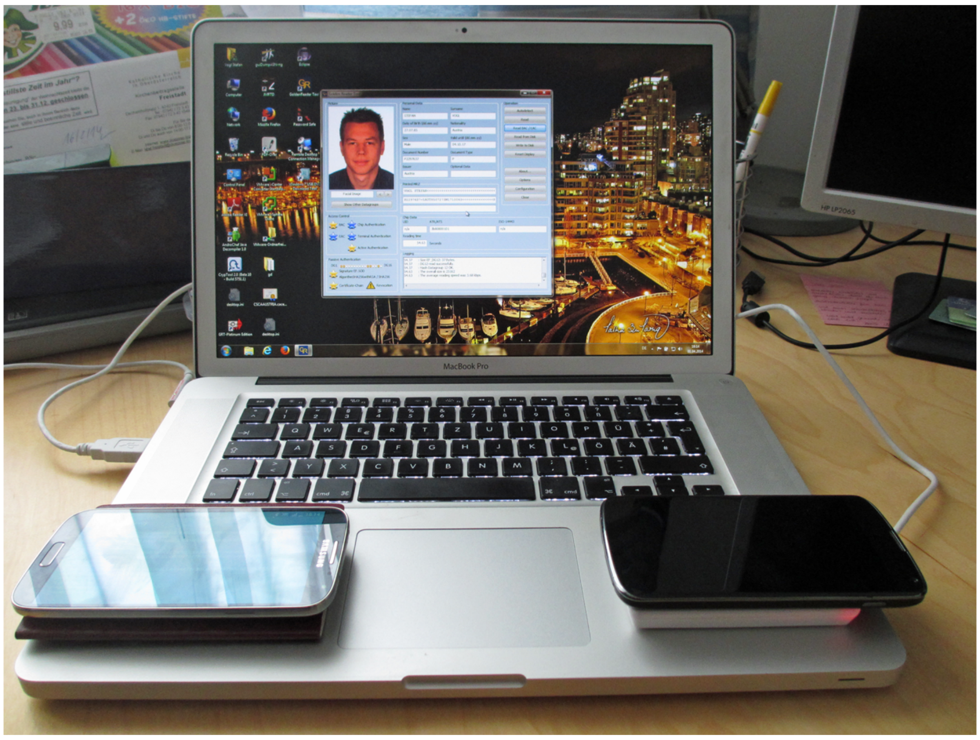


Figure 5.4: Setup during relay attack

and as the mole, we used a Samsung Galaxy S4 also running Android 4.4.2. Both mobile phones were connected via a WiFi Direct connection. The inspection system was replaced by a standard PC running the Golden Reader Tool by Secunet. This software uses the Secunet ePassport API[8], which is also used in real world inspection systems. Figure 5.4 shows the setup of our proof of concept.

[8]`http://www.secunet.com/en/topics-solutions/government/biometrics-eids/epassport-api/`

Due to the fact, that the proxy can only communicate with the reader on a very high level via APDUs, it is not possible to use low level commands, like the WTX command, to stall the reader, so that no timeouts are hit, if the mole cannot reply fast enough. Therefore, it depends very much on the latency of the network, which is used to connect the mole and the proxy if the attack is successful. In our test setup we used the peer to peer technology WiFi Direct, with which we never experienced any timeouts on the reader side.

The advantage of our implementation over the already presented attacks using specialized hardware is on the one hand the fact, that the proxy does not need another hardware acting as a host, but can be used as a standalone device. In [57], an NFC capable smartphone was used as a mole and as a controller for the proxy hardware using Bluetooth which makes an additional notebook unnecessary. In this case, the distance between the proxy and the mole is limited by the maximum distance of Bluetooth. Using two smartphones has the advantage that smartphones have Internet connectivity already integrated which makes the implementation of the communication between the proxy and the mole easier and works over far greater distances. The third point is, that NFC capable smartphones are widely available and can be bought cheaper than specialized hardware.

In [63], a similar approach was taken for implementing a relay attack. Here the proxy was implemented on a BlackBerry smartphone. Having the same advantages as using an Android smartphone, the BlackBerry platform lacks some of the advantages of Android. The most important one is that the presented attack can only be used with one explicit BlackBerry model whereas the relay attack we implemented can be used with a variety of Android smartphones.

Passport Simulator

The second function of the implemented Android application is an Android passport simulator. With this application it is possible to read a given passport and copy it to the smartphone's filesystem. The smartphone then simulates this passport to an external reader. If the copied passport uses some anti cloning protection like Active Authentication or Chip Authentication, the simulated passport can be detected by the inspection system. To avoid detection, the application makes use of the already described index manipulation attack. Datagroups indicating Active Authentication (DG15) or Chip Authentication (DG14) are stripped from the unprotected EF.COM. If the inspection system does not check the availability of these datagroups in the EF.SOD, which is signed and can therefore not be altered by the application without detection, the simulated passport will not be identified as a copy.

To read the data from the passport we can make use of the NFC Reader / Writer mode as we already did for implementing the mole in the relay attack. For sending the right APDUs to the passport and interpreting the responses we used the open source jMRTD[9] library. This library supports all necessary security features and is already prepared to make use of the Android NFC

[9]`http://jmrtd.org`

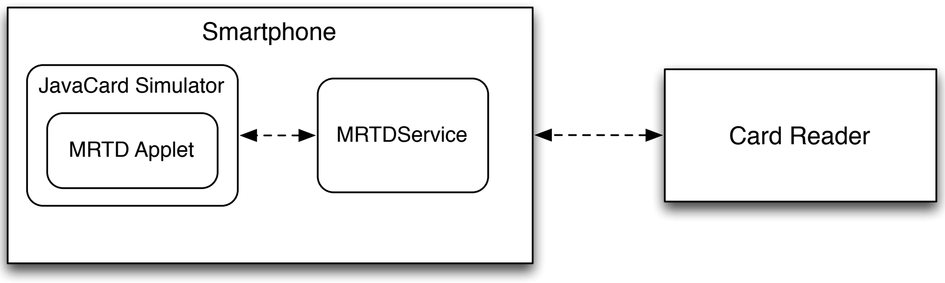


Figure 5.5: Overall architecture of the passport simulator.

stack. The fact that it is open source also helped to create a work around for a special case in Austrian passports. During development of this application we discovered, that in Austrian passports before 2011, the first name and the last name in DG11 are not separated using two delimiters, as suggested in the ICAO 9303 specifications, but only one. This made the processing of the passport data harder.

Once read, the data is stored on the phone. Each datagroup is saved to a file with its file identifier according to the specification as its filename. The resulting files are then compressed into a ZIP-file.

To simulate the saved passport to an external reader, the HostAPDUService from the relay attack described before was used. Instead of implementing a fully featured smartcard operating system, to be able to react to the APDUs from the external card reader, we again made use of the jMRTD library. They offer a JavaCard applet, which can be used to turn a standard JavaCard into an MRTD. This applet offers the possibility to first write an empty MRTD to a JavaCard and then use the *WRITE BINARY* commands to write the data to the JavaCard. Due to the fact, that the JavaCard applet expects a JavaCard virtual machine, we could not use the applet directly. To simulate a JavaCard where we could write the empty MRTD applet to, we used the open source JavaCard simulator jCardSim[10]. Originally the jCardSim API heavily depended on the Java SmartCardIO package. Upon our request, they also developed an interface to the simulator, that takes only byte arrays as an argument and is completely independent from the SmartCardIO package. This made it possible to use the simulator inside our HostAPDUService. Figure 5.5 shows the overall architecture of the implementation. Listing 5.4 shows how the JavaCard simulator is initialized using the jMRTD applet. Then the passport data is read from the filesystem and written to the JavaCard simulator.

To enable the jMRTD, to be as platform independent as possible, they make use of the Smart Card Utilities for Better Access (SCUBA). This library offers implementation for the Android NFC system and the SmartcardIO packages, which are used for all smartcard communication

[10]`http://jcardsim.org`

```
public void onCreate() {
  if (mode == MODE_SIMULATOR) {
    log.info("Running in simulator mode. Preparing JavaCard.");
    cardService = new SimulatorCardService();

    if (isExternalStorageReadable()) {
      try {
        org.jmrtd.PassportPersoService persoService = new
    PassportPersoService(
            cardService);
        persoService
            .burnPassport(new ZipFile(
                Environment.getExternalStorageDirectory()
                    + "/aMRTDSim/passports/7ec7dd63-8d82-4a77-b098-45
    de21b1d5ad.zip"));
      } catch (IOException e) {
        log.severe("Could not create passport from .zip file. "
            + e.getMessage());
      } catch (CardServiceException ce) {
        log.severe("Could not write passport to simulator. "
            + ce.getMessage());
      }
    } else {
      log.severe("Could not access external storage.");
    }
  } else if (mode == MODE_RELAY) {
    ...
  }
}
```

Listing 5.4: The onCreate() method for the passport simulator

in JavaSE applications. To be able to also use the JavaCard simulator as our communication endpoint instead of a real smartcard, we had to implement our own SCUBA CardService class. This class takes care of all communication to the simulator. As for the relay attack, the APDU that is sent from the reader to the smartphone, are processed in the *processCommandApdu(...)* method.

To test our implementation, we used the Golden Reader Tool Platinum Edition again. As already described, with the default settings, the index manipulation was not detected and the simulated passport was shown as genuine. If the settings are corrected and the EF.SOD is chosen as index for the available datagroups, the simulated passport is detected as a clone.

CHAPTER 6

Future Work and Conclusion

In this chapter, we describe the future work that will be conducted in the field of ePassport security and what current and future developments in the field of ePassports are. In the conclusion we summarize the findings of this thesis and show the direct implications of a secure passport system on all our lives.

6.1 Future Work and Future Developments

Future Work

One part of our future work is the optimization of the OpenCL implementation. We will conduct an analysis of the implementation of the SHA-1 and the 3DES and try to find which parts can be further optimized with an emphasis on GPU specific optimizations. This means, that we will try to exploit special features of GPUs like the very efficient calculations with vector types.

During our research and especially while reading [12], we found that the hardware implementation of the BAC cracker can be further optimized by using the pipelining technique for the SHA-1 part. This was not possible in their implementation because of the limitations of the Field Programmable Array (FPGA) they were using. Due to the fact, that the era of FPGAs used for Bitcoin mining is starting to be over, powerful FPGAs are available for a very low price. Although we believe, that it is not possible to fully pipeline the algorithm, even on these FPGAs, we suspect, that we can achieve a significant speed up compared to the implementation in [12]

Future Developments

Because of the weaknesses and multiple attacks on the Basic Access Control, the BSI developed PACE, as described in Section 3.5. This should make ePassports more resistant against skimming and eavesdropping attacks. The European Union decided, that this is a valid and dangerous threat, especially considering the long validity periods of passports, and therefore, the implementation of PACE is mandatory starting from the end of 2014.

Especially the research, that is done on ABC gates, shows the importance, but also the shortcomings of ePassports. As we can see in [64], one suggested topology for an ABC gate is the so called kiosk model. Here, the border control is done in two steps, whereas the passport is only needed in the first step, the kiosk, for a registration. For the second step, the border crossing itself, a token, issued in the first step, is used to link the user to his passport. This can go so far, as this first step could be done beforehand and can be valid for a longer time. Therefore, frequent travelers can leave their passports in their luggage while traveling. This shows, that although border control agencies still trust the security of the passport, they think, that the usability of handling ABC gates with a passport reader leaves room for improvement.

6.2 Conclusion

In the first part of this thesis, we conducted a thorough analysis of the Austrian ePassport implementation. We found that all in all two generations of ePassports can be identified. Between 2006 and 2009 no biometrics, except the photo, were stored in the passport. Therefore, no Extended Access Control was implemented. The Austrian implementation of these first generation ePassports included the mandatory security feature Passive Authentication, as well as the optional Basic Access Control, for access protection and the Active Authentication protocol for cloning protection. The second generation passports, which are in place since 2009, already contain two fingerprints of the passport holder. For additional protection of these fingerprints the Extended Access Control with Terminal Authentication for access protection and Chip Authentication for cloning protection was implemented. We also analyzed the chosen cryptographic algorithms like hashing algorithms, and found that all algorithms not specified by the ICAO were chosen very securely. This is very important when designing an ePassport ecosystem, because of the very long validity period. All the used algorithms have to be secure for the whole validity period of ten years.

In the second part we identified possible attack vectors and security threats to the ePassport. We found various threats, whereas the most dangerous are forgery, which means that an attacker can build an arbitrary ePassport dataset which can not be detected by the inspection system, and eavesdropping or skimming. Forgery, in our opinion, is the biggest threat to the entities that wants to verify the passport, like border guards, eavesdropping and skimming are a big privacy risk to the holder of the passport.

The third part consisted of a literature research, where we tried to identify already known attacks on the ePassport. During this research we implemented a bruteforce attack on the Basic Access Control keys to decrypt an eavesdropped encrypted communication using GPGPU programming. There we found that although we could not get a better performance than the FPGA implementation in [12], this implementation has the advantage that it can be easily distributed over multiple entity for example over the internet. In this case there are no additional hardware costs, because every PC and even smartphones already have a GPU installed, where most of them are already compatible with OpenCL and could be used for this project out of the box. Taking our results of 2^{22} calculated keys per second it would take 64 average PCs to achieve the 2^{28} keys per second that were achieve in [12]. As we have seen, it would take about 350,000 average PCs to crack the Basic Access Control keys in ten minutes. This may seem a lot, but considering the size of online voluntary clusters like the BOINC network, where millions of people calculate CPU intensive tasks together, 350,000 PCs are not unrealistic. We also found that, when comparing the amount of machines with botnets in underground markets, it would cost about 7,000 $ to buy a botnet, which is able to crack the Basic Access Control keys in 10 minutes.

One goal of this thesis was to show if it is possible to implement at least some of the found attacks on an Android smartphone. Especially because of the recently added Host Card Emulation functionality in Android 4.4, it was possible to implement an ePassport simulator using the jMRTD JavaCard applet and the JCardSim JavaCard simulator. Additionally we were able to successfully implement a relay attack with two Android smartphones, whereas only one of them has to support Host Card Emulation.

All in all, we can say, that the Austrian ePassport was designed very securely. Besides this, we also found, that there are still some open issues with the overall ePassport design, whereas most of the problems lie in the security of the implementations of the inspection systems. Therefore, it is very important for the whole ePassport system, that the inspection systems are implemented very carefully. As we already described earlier, there is a clear trend towards more and more ABC gates. Therefore, the printed security features are getting less important, whereas the importance of the electronic security features of the RFID chip is getting higher.

To guarantee the highest possible security, the European Union started the research project FastPass[1], which tries to develop a harmonized, modular approach to ABC gates. The project lays very much emphasis on security, data protection and ethics. There we can see, that the European Union takes the security issues found in inspection systems very seriously and that the future of border control lies in heavy automation.

One big advantage of ePassports is also, that the data within can be easily verified and compared to central databases and background systems. We already described the Austrian IDR, where all valid passports, including limitations, validity period and so on are centrally stored and can be easily accessed, not only by border guards, but also by all other police officers. There is also the possibility to look up checked passports in the Schengen Information System (SIS). This system holds information about people who are not allowed to enter or stay in the European Union, missing persons and certain property, such as passports for example. With the SIS, border guards are able to identify an identity document, such as a passport, as stolen or lost. Another international database containing lost and stolen travel documents is the Interpol Stolen and Lost Travel Documents Database (SLTD). The need to make such checks more common during border control, was recently shown during the plane crash of the Malaysian Airline flight

[1]`https://www.fastpass-project.eu/`

MH370. One Austrian and one Italian passenger were listed on the flight, whereas none of them really were. Both claimed to have their passport stolen in Thailand during the last two years. Although both passports could already be found in the Interpol SLTD database, the Malaysian authorities did not check it during the border crossing procedure. If the checks had been made, the two passengers, who were suspected to be terrorists, could have been stopped from boarding the plane [65].

If the two passengers, with the stolen passports, hadn't stolen the documents, but were able to clone them, without the owners noticing it, the passports wouldn't have been in the central database and the border control would have had no possibility to verify the passport. This shows, that a protection against cloning is very important, even if these attacks are often said to be impractical, because the picture in the passport can't be changed by the attacker and can therefore only be used for so called look-a-like attacks. As we can see in the example of the two stolen passports, this is a valid threat.

APPENDIX A

Detailed Error Codes

Table A.1 shows the detailed description of the possible statuswords in an Response-APDU and their meaning.

SW1	SW2	Meaning
’62’	’00’	No information given
	’02’ to ’80’	Triggering by the card.
	’81’	Part of returned data may be corrupted
	’82’	End of file or record reached before reading N_e bytes
	’83’	Selected file deactivated
	’84’	File control information not formatted according to ISO 7816-4 5.3.3
	’85’	Selected file in termination state
	’86’	No input data available from a sensor on the card
’63’	’00’	No information given
	’81’	File filled up by the last write
	’CX’	Counter from 0 to 15 encoded by ’X’
’64’	’00’	Execution error
	’01’	Immediate response required by the card
	’02’ to ’80’	Triggering by the card
’65’	’00’	No information given
	’81’	Memory failure
’68’	’00’	No information given
	’81’	Logical channel not supported
	’82’	Secure Messaging not supported
	’83’	Last command of the chain expected
	’84’	Command chaining not supported
’69’	’00’	No information given
	’81’	Command incompatible with file structure
	’82’	Security status not satisfied
	’83’	Authentication method blocked
	’84’	Reference data not usable
	’85’	Conditions of use not satisfied
	’86’	Command not allowed
	’87’	Expected SM data objects missing
	’88’	SM data objects incorrect
’6A’	’00’	No information given
	’80’	Incorrect parameters in the data field
	’81’	Function not supported
	’82’	File not found
	’83’	Record not found
	’84’	Not enough memory space in the field
	’85’	L_c inconsistent with TLV structure
	’86’	Incorrect parameters P1-P2
	’87’	L_c incosistent with P1-P2
	’88’	Referenced data not found

Table A.1: Warning and error conditions [24]

APPENDIX B

Analysed Passports

Table B.1 shows the document numbers, date of issuance and date of expiry of the analysed passports.

Document Number	**Date of Issuance**	**Date of Expiry**
P1077407	05.07.2006	04.07.2016
P1149270	27.07.2006	26.07.2016
P1324430	09.10.2006	08.10.2016
P1830267	24.04.2007	23.04.2017
P2166795	26.07.2007	25.07.2017
P2297637	05.10.2007	04.10.2017
P2801355	05.06.2008	04.06.2018
P2803214	05.06.2008	04.06.2018
P2819776	13.06.2008	12.06.2018
P3204147	19.12.2008	18.12.2018
P3343970	30.03.2009	29.03.2019
P3754850	03.08.2009	02.08.2019
P3958583	17.11.2009	16.11.2019
P4586504	14.05.2010	13.05.2020
P5922332	21.09.2011	20.09.2021
P6360459	03.05.2012	02.05.2022

Table B.1: List of analysed passports

Bibliography

[1] International Civil Aviation Organization (ICAO). *Machine Readable Travel Documentes (MRTDs): History, Interoperability, and Implementation*. 2007. URL: http://www.icao.int/Security/mrtd/Downloads/Technical%20Reports/ICAO_MRTD_History_of_Interoperability.pdf (visited on 07/25/2014).

[2] Brenda McPhail, Christopher Parsons, Karen L. Smith, et al. "Identifying Canadians at the Border: ePassports and the 9/11 Legacy". In: *Canadian Journal of Law and Society* 27.3 (2012), pp. 341–361.

[3] European Council. *Council Regulation (EC) No 126/2006*. URL: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:403:0018:0060:EN:PDF (visited on 06/20/2014).

[4] European Commission. *Commission Regulation (EU) No 383/2012 of 4 May 2012 laying down technical requirements with regard to driving licences which include a storage medium (microchip)*. URL: http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=OJ:L:2012:120:TOC (visited on 01/12/2014).

[5] International Civil Aviation Organization (ICAO). *Machine Readable Travel Documents, Doc 9303*. 2006. URL: http://www.icao.int/publications/pages/publication.aspx?docnum=9303 (visited on 07/25/2014).

[6] European Commission. *Commission Directive 2003/127/EC of 23 December 2003 amending Council Directive 1999/37/EC on the registration documents for vehicles*. URL: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2004:010:0029:0053:EN:PDF (visited on 01/12/2014).

[7] Barbara Wimmer. *Alle Bankomatkarten bekommen NFC-Funktion*. 2013. URL: http://futurezone.at/produkte/alle-bankomatkarten-bekommen-nfc-funktion/24.591.161 (visited on 08/04/2014).

[8] Michael Roland and Josef Langer. "Cloning Credit Cards: A Combined Pre-play and Downgrade Attack on EMV Contactless". In: *7th USENIX Workshop on Offensive Technologies, Berkeley and CA: USENIX, 2013*. 2013.

[9] Ari Juels, David Molnar, and David Wagner. "Security and Privacy Issues in E-passports". In: *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*. IEEE, 2005, pp. 74–88. ISBN: 0-7695-2369-2.

[10] Gaurav S. Kc and Paul A. Karger. *Preventing Attacks on Machine Readable Travel Documents (MRTDs)*. 2005. URL: http://eprint.iacr.org/2005/404 (visited on 07/25/2014).

[11] Lukas Grünwald. *Security by Politics - Why it will never work*. 2007. URL: http://www.defcon.org/images/defcon-15/dc15-presentations/dc-15-grunwald.pdf (visited on 07/25/2014).

[12] Yifei Liu, Timo Kasper, Kerstin Lemke-Rust, et al. *E-Passport: Cracking Basic Access Control Keys with COPACOBANA*. 2007. URL: http://emsec.ruhr-uni-bochum.de/media/crypto/veroeffentlichungen/2011/01/29/epasscrack_sharcs_2007.pdf (visited on 08/01/2014).

[13] Jeroen van Beek. *ePassports reloaded*. 2008. URL: https://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-vanBeek/BlackHat-Japan-08-Van-Beek-ePassports.pdf (visited on 11/03/2013).

[14] Henning Richter, Wojciech Mostowski, and Erik Poll. “Fingerprinting Passports”. In: *NLUUG 2008 Spring Conference on Security*. 2008, pp. 21–30. URL: http://www.cs.ru.nl/~erikpoll/papers/nluug.pdf (visited on 01/02/2014).

[15] Tom Chothia and Vitaliy Smirnov. “A Traceability Attack Against e-Passports”. In: *14th International Conference on Financial Cryptography and Data Security FC10*. 2010, pp. 1–15.

[16] Panagiotis Papantonakis, Dionisios Pnevmatikatos, Ioannis Papaefstathiou, et al. “Fast, FPGA-based Rainbow Table creation for attacking encrypted mobile communications”. In: *2013 23rd International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–6.

[17] Michael B. Taylor. *Bitcoin and the Age of Bespoke Silicon*. 2013. URL: http://cseweb.ucsd.edu/~mbtaylor/papers/bitcoin_taylor_cases_2013.pdf (visited on 08/01/2014).

[18] Passenger Terminal Today. *Vision-Box installs 24 self-boarding gates at Lisbon International*. 2013. URL: http://www.passengerterminaltoday.com/viewnews.php?NewsID=53818 (visited on 06/28/2014).

[19] N. Kose and J. L. Dugelay. “On the vulnerability of face recognition systems to spoofing mask attacks”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. 2013, pp. 2357–2361.

[20] Klaus Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. 3rd ed. Chichester: Wiley & Sons, 2010. ISBN: 0470665122.

[21] Mark Roberti. “The History of RFID Technology”. In: (2005). URL: http://www.rfidjournal.com/ (visited on 12/07/2013).

[22] Flavio D. Garcia, Gerhard Koning Gans, Ruben Muijrers, et al. “Dismantling MIFARE Classic”. In: *Computer Security - ESORICS 2008*. Ed. by Sushil Jajodia and Javier Lopez. Vol. 5283. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 97–114. ISBN: 978-3-540-88312-8.

[23] International Organization for Standardization. *ISO/IEC 14443, Identification cards – Contactless integrated circuit cards – Proximity cards*. 2008.

[24] International Organization for Standardization. *ISO/IEC 7816, Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange*. 2013.

[25] International Telecommunication Union. *Information Technology — Abstract Syntax Notation One (ASN.1): Specification of Basic Notation*. 2002.

[26] Bundesamt für Sicherheit in der Informationstechnik. *Nach Technischen Richtlinien zertifizierte Produkte: Elektronische Pässe & Ausweise*. 2014. URL: `https://www.bsi.bund.de/DE/Themen/ZertifizierungundAnerkennung/ZertifizierungnachTR/ZertifizierteProdukte/zertifizierteprodukte_node.html` (visited on 04/01/2014).

[27] NXP Semiconductors. “P5CD080 - Secure Dual Interface PKI Smart Card Controller: Short Form Specification”. In: (2008).

[28] NXP Semiconductors. *U.S. State Department Advances NXP Technology for ePassport Program*. 2006. URL: `http://www.nxp.com/news/press-releases/2006/09/u-s-state-department-advances-nxp-technology-for-epassport-program.html` (visited on 07/25/2014).

[29] NXP Semiconductors. *P5CD072 - Secure Dual Interface PKI Smart Card Controller: Short Form Specification*. 2004.

[30] International Organization for Standardization. *ISO/IEC 9796-2:2010 - Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms*.

[31] Bundesamt für Sicherheit in der Informationstechnik. *Advanced Security Mechanisms forMachine Readable Travel Documents - Part 1*. 2012. URL: `http://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/TR-03110%5C_v2.1%5C_P2pdf.pdf?%5C_%5C_blob=publicationFile` (visited on 07/25/2014).

[32] Bundesamt für Sicherheit in der Informationstechnik. *Worked Example for Extended Access Control (EAC)*. 2010. URL: `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/BSI_EAC-Worked-Example.zip` (visited on 07/25/2014).

[33] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1996. ISBN: 0-471-11709-9.

[34] International Organization for Standardization. *ISO/IEC 14443, Identification cards – Contactless integrated circuit cards – Proximity cards - Part 2: Radio frequency power and signal interface*. 2010.

[35] International Civil Aviation Organization (ICAO). *Regulations for the ICAO Public Key Directory*. 2011. URL: http://www.icao.int/Security/mrtd/PKD%20Documents/Regulations%20for%20the%20ICAO%20PKD.pdf (visited on 07/25/2014).

[36] Roman Vanek. "The ICAO PKD State of Play". In: *MRTD Report* 8.2 (2013), pp. 4–7. URL: http://www.icao.int/publications/journalsreports/2013/MRTD_Report%20_Vol8_No2.pdf (visited on 07/25/2014).

[37] Bundesamt für Sicherheit in der Informationstechnik. *Common Certificate Policy for the Extended Access Control Infrastructure for Passports and Travel Documents Issued by EU Member States*. 2013. URL: http://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03139/BSI-TR-03139%5C_pdf.pdf?%5C_%5C_blob=publicationFile (visited on 07/25/2014).

[38] Ceska Technicka Norma. *CSN 36 9791 - Information technology – Country Verifying Certification Authority Key Management Protocol for SPOC*. 2009. URL: http://nahledy.normy.biz/nahled.php?i=85000 (visited on 07/22/2014).

[39] Österreichisches Parlament. *Passgesetz 1992*.

[40] Elaine Barker, William Barker, William Burr, et al. *Recommendation for Key Management – Part 1: General (Revision 3)*. 2012. URL: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57%5C_part1%5C_rev3%5C_general.pdf (visited on 07/25/2014).

[41] National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. 2012. URL: http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf (visited on 07/25/2014).

[42] Jean-Sebastien Coron, David Naccache, Mehdi Tibouchi, et al. *Practical Cryptanalysis of ISO/IEC 9796-2 and EMV Signatures*. 2009. URL: http://eprint.iacr.org/2009/203 (visited on 07/25/2014).

[43] Internet Engineering Task Force. *RFC 5639: Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*. 2010. URL: http://www.ietf.org/rfc/rfc5639.txt.

[44] European Commission. *Commission Decision of 4.8.2011*. URL: http://ec.europa.eu/dgs/home-affairs/e-library/docs/comm_native_c_2011_5499_f_en.pdf (visited on 01/12/2014).

[45] Frontex Agency. *Operational and Technical security of Electronic Passports*. 2011. URL: http://frontex.europa.eu/assets/Publications/Research/Operational_and_Technical_Security_of_Electronic_Pasports.pdf (visited on 07/22/2014).

[46] Florian Pfeiffer, Klaus Finkenzeller, and Erwin Biebl. "Theoretical Limits of ISO/IEC 14443 type A RFID Eavesdropping Attacks". In: *Smart Objects, Systems and Technologies (SmartSysTech), Proceedings of 2012 European Conference on*. 2012, pp. 1–9.

[47] Bundesamt für Sicherheit in der Informationstechnik. *Messung der Abstrahleigenschaften von RFID-Systemen (MARS)*. 2008. URL: https://www.bsi.bund.de/DE/Themen/ElektronischeAusweise/RadioFrequencyIdentification/MARSStudie/marsstudie.html (visited on 07/23/2014).

[48] Leslie A. Zebrowitz, Joann Montepare, and Michael D. Lee. “They don’t all look alike: individuated impressions of other racial groups”. In: *Journal of personality and social psychology* 65.1 (1993), pp. 85–101. ISSN: 0022-3514. URL: http://view.ncbi.nlm.nih.gov/pubmed/8355144 (visited on 07/25/2014).

[49] Bundesministerium für Inneres. *Identitätsdokumenteregister*. 2007. URL: http://www.digitales.oesterreich.gv.at/DocView.axd?CobId=22257 (visited on 07/23/2014).

[50] Ludovic Rousseau. *Smart card ATR parsing*. 2009. URL: http://smartcard-atr.appspot.com/ (visited on 07/23/2014).

[51] Helena Handschuh and Bart Preneel. *On the Security of Double and 2-key Triple Modes of Operation*. 1999. URL: http://cryptoshop.com/de/downloads/hp99doub2keytripledes.pdf (visited on 07/24/2014).

[52] NVIDIA. “CUDA Programming Guide”. In: (2013). URL: http://www.nvidia.com/content/cudazone/download/OpenCL/NVIDIA_OpenCL_ProgrammingGuide.pdf (visited on 07/25/2014).

[53] Khronos. *OpenCL Specification*. 2012. URL: https://www.khronos.org/registry/cl/specs/opencl-1.2.pdf (visited on 07/25/2014).

[54] Internet Engineering Task Force. *RFC 3174 - US Secure Hash Algorithm 1 (SHA-1)*. 2001. URL: http://tools.ietf.org/html/rfc3174.

[55] Brian Donohue. *How Much Does a Botnet Cost*. 2013. URL: http://threatpost.com/how-much-does-botnet-cost-022813/77573 (visited on 07/25/2014).

[56] Frontex Agency. *Biopass II: Automated biomteric border crossing systems based on electronic passports and facial recognition: RAPID and SmartGate*. 2010. URL: http://frontex.europa.eu/assets/Publications/Research/Biopass_Study_II.pdf (visited on 10/12/2013).

[57] Wolfgang Issovits and Michael Hutter. “Weaknesses of the ISO/IEC 14443 protocol regarding relay attacks”. In: *RFID-Technologies and Applications (RFID-TA), 2011 IEEE International Conference on*. 2011, pp. 335–342. ISBN: 978-1-4577-0028-6. DOI: 10.1109/RFID-TA.2011.6068658.

[58] Gerhard P. Hancke and Markus G. Kuhn. *An RFID Distance Bounding Protocol*. 2005. URL: http://www.cl.cam.ac.uk/~mgk25/sc2005-distance.pdf (visited on 08/01/2014).

[59] Rishab Nithyanand. *A Survey on the Evolution of Cryptographic Protocols in ePassports*. 2009. URL: https://eprint.iacr.org/2009/200.pdf (visited on 08/01/2014).

[60] Jeroen van Beek. *THC-ePassports*. 2008. URL: https://www.thc.org/thc-epassport/ (visited on 10/27/2013).

[61] International Organization for Standardization. *ISO/IEC 18092:2013 - Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1)*. 2004. (Visited on 10/13/2013).

[62] Google Inc. *Android Developer Guide*. 2013. URL: http://developer.android.com/guide/index.html (visited on 01/06/2014).

[63] Lishoy Francis, Gerhard Hancke, Keith Mayes, et al. *Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones*. 2011. URL: https://eprint.iacr.org/2011/618.pdf (visited on 07/25/2014).

[64] Frontex Agency. *Best Practice Operational Guidelines for Automated Border Control (ABC) Systems*. 2012. URL: http://frontex.europa.eu/assets/Publications/Research/Best_Practice_Operational_Guidelines_for_Automated_Border_Control.pdf.

[65] Austrian Press Agency. *Flug MH370: Malaysia laut Interpol bei Reisepass-Kontrollen nachlässig*. 2014-03-29. URL: http://derstandard.at/1395363596792/Flug-MH370-Malaysia-laut-Interpol-bei-Reisepass-Kontrollen-nachlaessig (visited on 07/25/2014).