UNIVERSITÄT PADERBORN

Die Universität der Informationsgesellschaft

Faculty of Electrical Engineering, Computer Science and Mathematics

Master's Thesis

An Analysis of Traceability of Electronic Identification Documents

Student:	**Frederik Möllers**
Field of Study:	**Computer Science**
Advisor:	**Jun.-Prof. Dr. Christoph Sorge**
Second Advisor:	**Prof. Dr. rer. nat. Johannes Blömer**

Abstract

While electronic identification documents featuring contactless interfaces become more and more common, the concern about the security of these devices grows proportionally. Identity theft, fake identities and traceability of people are some of the dangers that come with the introduction of this new technology. While the parties involved in the development try to make the documents as secure as possible, many people doubt that this will keep adversaries from exploiting the devices for their own benefit.

This thesis provides an objective analysis of the susceptibility of electronic identification documents to traceability attacks, using the new German Identification Card as an example. Its features to prevent tracking of individuals are examined and tested to determine whether they comply to the high requirements that are needed to keep their owners safe from malicious parties. Existing software is used and extended during the work and later utilized to uncover a weakness in the response behaviour of the RFID chip that is included in these documents.

Using this information, attackers are able to recognize the card of a previously observed victim within a set of several hundred similar cards. This poses a great threat to the privacy of the users, as in a few years almost all German citizens will carry such an electronic Identification Card.

Acknowledgements

I would firstly like to thank Sebastian Seitz for his support, guidance and helpful advice during my work.

Just as much would I like to thank Holger Funke of HJP Consulting for his advice and support. Without the knowledge and material he provided I would not have been able to finish this work in its extent within the five months.

Additional thanks go to the persons who lent me their Identification Cards for testing purposes: Aaron Bamberg, Jens Bewermeier, Tim Hartung, Dr. Jan Möllers, Jun.-Prof. Dr. Christoph Sorge as well as the students of the lectures "Datenschutz" and "Grundlagen der Programmierung 1".

Finally I would like to thank my family and friends for all their invaluable support.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

In November 2010, the German Ministry of the Interior introduced the new electronic identification card [14]. Several new security features were added [4, 19] to improve those already present in the electronic passport. The purpose of these new features was to increase the level of security and to mitigate possible attacks like those mounted on the passport [9, 28, 11].

While the protocols and standards used by the electronic identification card are considered to provide adequate security against compromisation of personal data, there has yet been little to no research on tracking and recognition of previously "seen" cards. Previous research has focused on different RFID devices, analyzing aspects like electrical [12] and timing properties [33].

1.2 Scope of Work

This thesis addresses the problem of traceability for electronic travel documents such as the new electronic identification card by using side-channel information. In the scope of this work, parts of the system such as random number generators, protocol implementations, timings and other characteristics of the

employed RFID chips are analyzed for weaknesses that could be exploited.

For example, predictability of random numbers used in the communication protocols would allow an adversary to determine wether or not he has seen a certain device before. Distinct behavior of chips produced by a certain manufacturer (e.g. differences in timing or different reactions to invalid input) would open the possibility to distinguish between classes of devices and might ultimately lead to the ability of matching device characteristics to persons.

For all scenarios, it is assumed that the attacker does not have physical access to the card by visual or mechanical means. This would already allow him to distinguish between cards using the data printed on them. The only feature assumed to be consistently available is the opportunity to initiate a wireless communication. In addition to this, the adversary might be able to acquire prior knowledge about the card owner, such as the name, birthdate, card issue date or similar data. Wether or not this prior knowledge is needed in a certain case will be determined and the preliminary conditions for each attack vector will be collected.

The work on this topic will be done using real hardware to make the scenario as realistic as possible and to ensure that no properties of a physical environment (as opposed to a virtual one) are missed during the analysis. This also introduces the risk of measurement errors and deficiencies of the employed hardware. Due to the examination being focused on the practicability of attacks in the real world though, the benefits outweigh the risks.

1.3 Approach

The work can be divided into three parts. The first part will be to determine which characteristics of the examined travel documents should be collected for analysis and how the collection as well as the analysis can be done. The random numbers used by the card during identification and for the setup of an encrypted connection provide a reasonable starting point. Their entropy can be calculated using different existing methods. Timing information is a second candidate. It is possible to measure wether different devices show distinct behavior in terms of calculation periods and response times.

The second part will consist of the implementation of a system to gather the data specified in the first part. This implementation will be based on the Proxmark III RFID Tool [34]. An open-source firmware for the device is readily available and can be customized to accomodate the needed functionality.

Once enough data has been gathered, it will be analyzed by different means as the third part of the work. The goal of this part is to find out wether there are characteristics in the devices that enable an adversary to track a single document over longer periods of time. The dieharder test suite [7], for example, can be used to examine aforementioned random numbers.

Chapter 2

Terms, Definitions and Protocols

This chapter covers the basic terms, definitions and protocols which are used and referenced throughout the entire thesis. At first, technical terms are explained or defined and after that the operations of protocols are given.

2.1 Terms and Definitions

In the following paragraphs terms that are used throughout the thesis and especially in the protocol descriptions in chapter 2.2 are explained and defined.

2.1.1 APDU

An APDU (**A**pplication **P**rotocol **D**ata **U**nit) as defined in the ISO-7816 standard [27] is a packet used in the communication between smartcards like the new German Identification Card and a reader device. There are two classes of APDUs: Command APDUs and Response APDUs.

- Command APDUs are sent by the reader and consist of a 4-byte header, an optional data body preceded by a byte indicating the length and an optional byte telling the expected length of the Response APDU. The

unit for the length specifications is bytes.

- Response APDUs are sent by the card as answers to Command APDUs and consist of an optional data part followed by a 4-byte status word. Response APDUs do not include a header.

Figure 2.1 shows the structure of Command and Response APDUs.

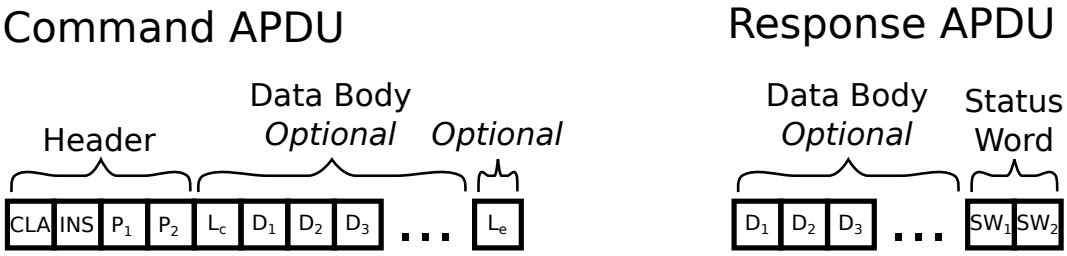


Figure 2.1: *The structure of APDUs.*
Abbreviations: CLA — Class; INS — Instruction; P_1 — Parameter 1; P_2 — Parameter 2; L_c — Command Length; $D_1, D_2, D_3, ...$ — Data Byte 1, 2, 3, ...; L_e — Expected Response Length; SW_1, SW_2 — Status Word Byte 1, 2

2.1.2 CAN

The CAN (**C**ard **A**ccess **N**umber) is a document-specific password printed on the new German Identification Card. It can be used as a password for the PACE protocol (see section 2.2.2), a password-based key exchange for encrypted communication. Only governmental authorities like customs and police have the possibility to read the content of cards using the CAN as a password for PACE. Figure 2.2 shows the location of the CAN on the card.

2.1.3 EAC

The term EAC (**E**xtended **A**ccess **C**ontrol) is used to group a set of protocols and security measures that protect electronic identification documents from unauthorized access. For the German electronic Identification Card these are Chip Authentication (see section 2.2.4) and Terminal Authentication (see section 2.2.3).

Figure 2.2: *The CAN (marked in red) on the new German Identification Card.*

In the case of the German electronic Identification card, a list of the supported protocols and their versions can be read from the chip by any capable device without authorization.

2.1.4 MRZ

The MRZ (**M**achine **R**eadable **Z**one) is a section on the identification document which includes distinctive information in a format that is easily readable by a machine using OCR software. It usually includes the name of the owner, the document number and other information depending on the type of document. Figure 2.3 shows the location of the MRZ on the back of the new German Identification Card.

For the PACE protocol (see section 2.2.2), a password can be computed by applying a *Key Derivation Function* [19] to specific parts of the MRZ. Similarly to the CAN, only governmental authorities can read data from the card after executing PACE with the derived password.

2.1.5 PIN/PUK

The PIN (**P**ersonal **I**dentification **N**umber) is a 6-digit code given to the owner of an identification card. It is used—similarly to the CAN—as a password for

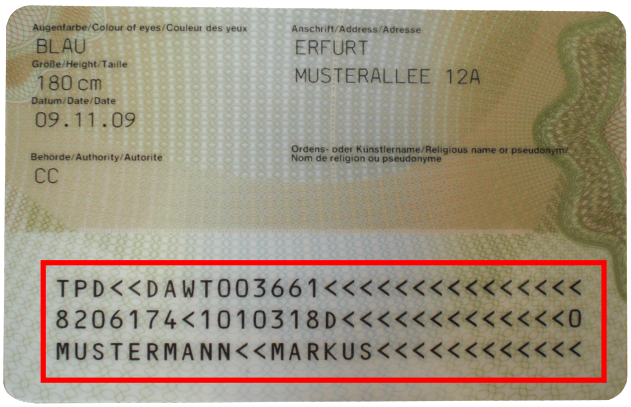


Figure 2.3: *The MRZ (marked in red) on the back of the new German Identification Card.*

the PACE key exchange (see section 2.2.2). In contrast to the MRZ and the CAN, the execution of PACE with the PIN serves as an initialization for private use, such as authorization in online shops. The number is not printed anywhere on the card to make sure that a stolen or lost card cannot be abused.

If the user enters the wrong PIN during the authentication procedure 3 times in a row, the card will lock up and prevent any further tries unless it is unlocked with the PUK (**P**ersonal **U**nblocking **K**ey). The PUK is a 10-digit code similar to the PIN, but the increased length increases its strength against a brute force cracking attempt.

2.1.6 UID/PUPI

In the context of the ISO-14443 communication protocols [23, 24, 25, 26], a **UID** (**U**nique **Id**entifier) or **PUPI** (**P**seudo-**U**nique **P**roximity Card **I**dentifier) is a (pseudo-) unique device identifier which distinguishes one RFID card from others in a set. It is used to address a single card among several others in the vicinity of a single reader. In the case of the German Identification Card, this identifier is to be generated randomly every time the card enters the field of a reader.

2.2 Protocols

This section describes the protocols that are supported by the German Identification Card. They are ordered by the sequence in which they are executed during a typical use case scenario. These use cases are detailed in chapter 3.1.

2.2.1 ISO-14443 Card Select

Before high-level protocols between an RFID reader and a card can be set up, the two devices have to establish a communication link between each other. This makes sure the reader can distinguish between multiple different cards in its vicinity and a card only responds to commands which are meant for it specifically [25]. For this scenario the ISO standard assumes that only one RFID reader is communicating with a set of cards in its vicinity.

In an anticollision sequence [25], the reader probes the UIDs or PUPIs of each card in its vicinity (the exact process is described in the standard, but is unimportant for this thesis). After that, it assigns a *Card Identifier* (CID) to each card it will communicate with by issuing a special selection command which includes the UID/PUPI of that card together with the CID. Higher-level protocols always include the CID in packets, so that a card can determine wether it should react to a packet or not. Figure 2.4 shows the procedure of establishing a link between the reader R and a card B.

2.2.2 PACE

PACE (**P**assword **A**uthenticated **C**onnection **E**stablishment [19] is a protocol for the establishment of a secure messaging channel between an RFID reader and the RFID chip of an electronic identification document. Its purpose is to make sure that only persons with visual access to the card or knowledge of the PIN can initiate a communication with it and to make eavesdropping on the data transfer infeasible.

The protocol serves as an improved alternative to the relatively weak [28] BAC (**B**asic **A**ccess **C**ontrol) [22]. In advantage to BAC, PACE uses ephemeral ses-

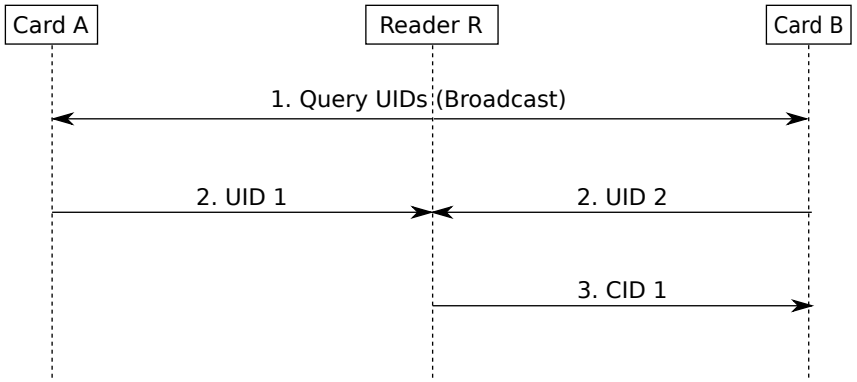


Figure 2.4: *A simplified illustration of the ISO-14443 Card Select procedure. Step 2 actually consists of multiple sub-steps to avoid collisions between simultaneously transmitted UIDs of different cards.*

sion keys that are sufficiently strong although the complexity of the password might be as low as a 6-digit code. This makes an offline brute force attack against the password impossible without cracking the much more complex session keys first.

The PACE protocol is based on the Diffie–Hellman key exchange [19, 16] and figure 2.5 shows a simplified illustration of an execution of PACE. Since the key exchange alone is susceptible to a man-in-the-middle attack, the PACE protocol incorporates a modification in the initial step. The card generates a random number which is used by both the card as well as the terminal to derive the ephemeral keys from which later the shared secret (the symmetric key) is computed. This random number is encrypted with a password before being passed to the reader. The password can be either derived from the MRZ (see section 2.1.4), it can be the CAN (see section 2.1.2) or the PIN (see section 2.1.5. Thus, an adversary would have to know this password to be able to mount a man-in-the-middle attack, as it would have to use the same seed as the card for the key generation.

This modification also lowers the risk of a password cracking attack, because an adversary has no means to decide wether a password used for the decryption of the initially generated random number is correct unless he uses it to execute the protocol and either fails or succeeds. Hence, he would theoretically on average have to make $\frac{\text{Number of possible passwords}}{2}$ attack attemps before finding the correct password. Counting only the card's response times of an attempt

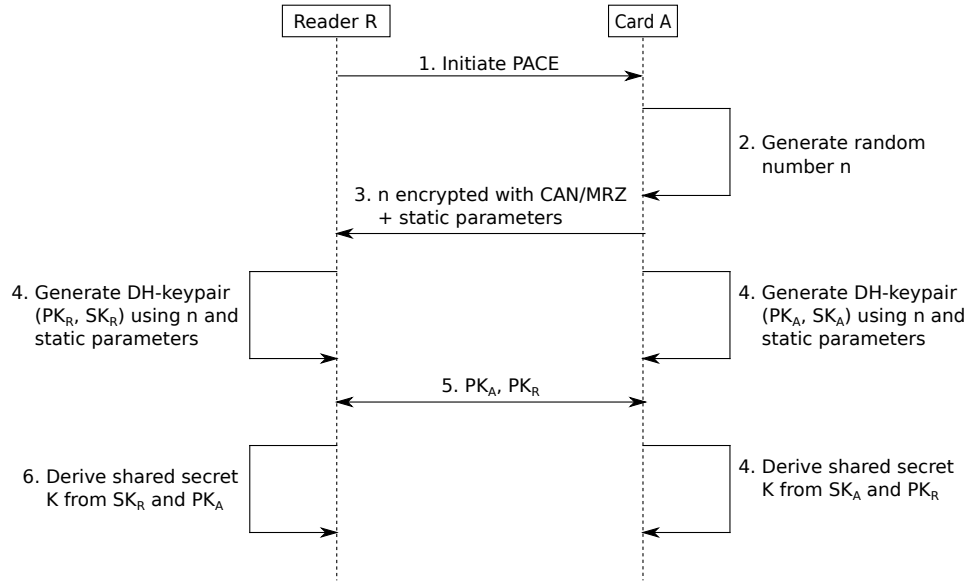


Figure 2.5: *A simplified illustration of the PACE procedure.*

using existing software [20], one try takes at least 1.2 seconds (not counting the computational effort), so a lower bound for cracking the password is thus approximately 7 days. If the PIN is used, the card will also lock up after 3 failed tries, preventing any further authentication attemps unless it is unlocked with the PUK (see section 2.1.5).

PACE alone however does not provide protection against an attacker who has knowledge of the password, e.g. by managing to get a look at the document itself.

2.2.3 Terminal Authentication

In order to guarantee that a Identification Card reader is entitled to access certain information stored on the card, it has to authenticate itself. There are two versions of a protocol for Terminal Authentication [19]. The common procedure in both versions is that the reader sends a certificate chain to the card which the card verifies using the saved public key of the topmost certification authority. The card then sends a random number n to the reader. On the reader, the number is concatenated with additional data and signed with the

reader's private key SK_R. The signature is sent back to the card and is checked using the reader's public key PK_R extracted from the certificate chain.

The two protocol versions differ in the additional data which is also signed by the reader. Version 2 was designed in a way that ties it to PACE and Chip Authentication so that the three protocols build on each other. Figure 2.6 shows the general procedure of this version which is implemented on the new German Identification Card.

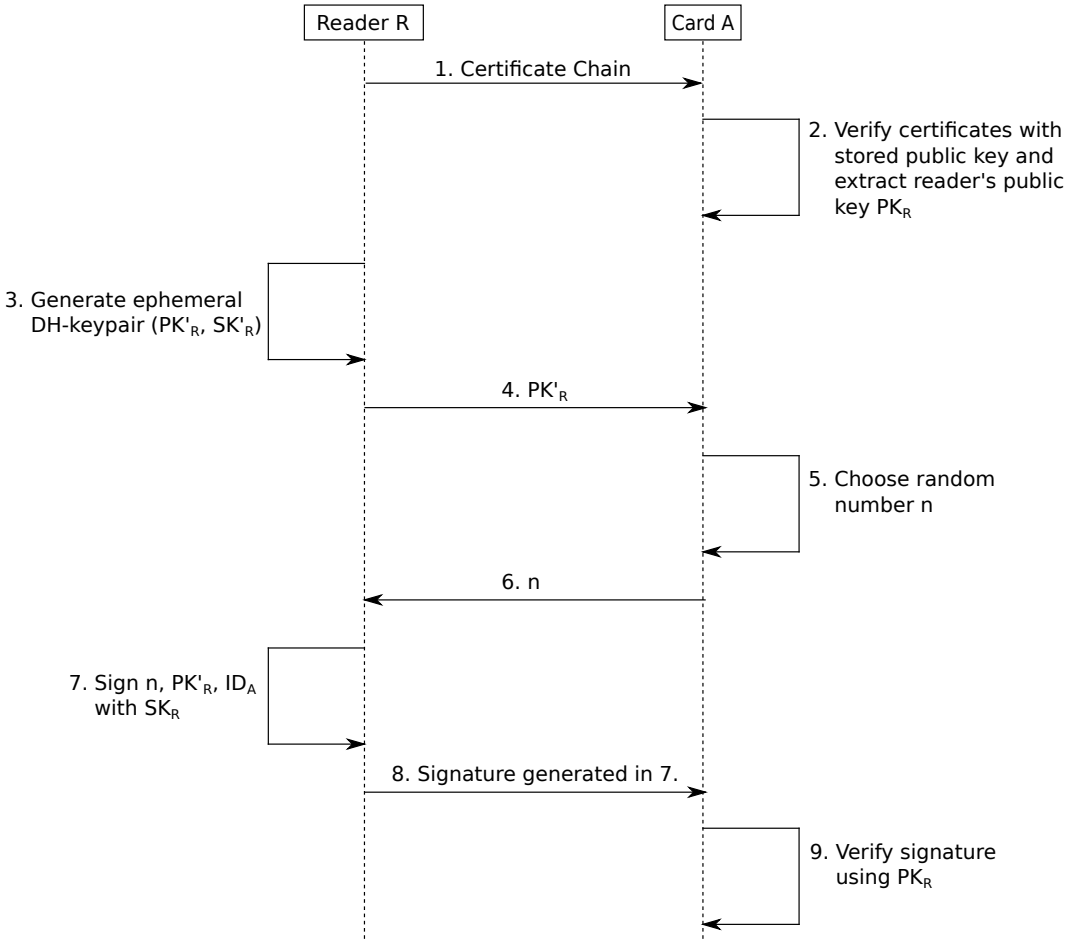


Figure 2.6: *A simplified illustration of the Terminal Authentication version 2 procedure.*

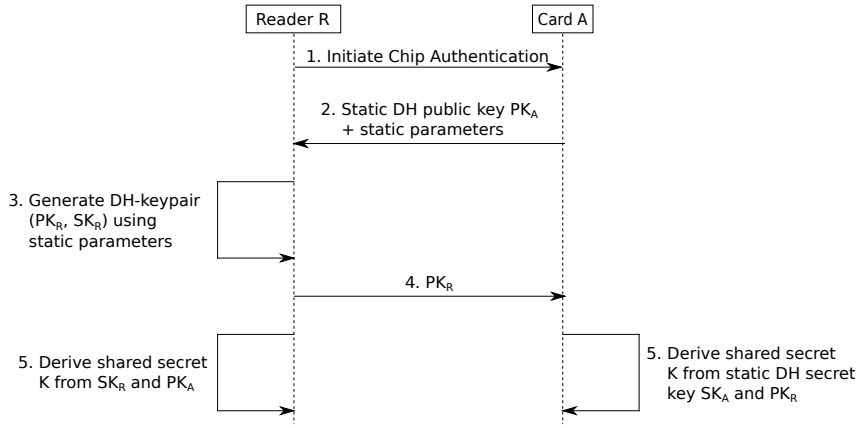


Figure 2.7: *A simplified illustration of the Chip Authentication procedure. In version 2 of the protocol, the ephemeral keys of the reader are not generated in step 3. Instead, the keys generated during Terminal Authentication are used. After this key exchange, the terminal can read the card key's signature and verify that the document is authentic.*

2.2.4 Chip Authentication

Chip Authentication as defined in the technical guideline TR-03110 [19] provided by the BSI is little more than a Diffie–Hellman key exchange [16]. The card uses a static key pair and the reader uses an ephemeral key pair either generated on the fly or taken from Terminal Authentication (see section 2.2.3), depending on the protocol version. The public key of the card is signed by a federal authority and this signature is saved on the chip, thus the reader can verify that the card is a valid, authentic document by reading and verifying said signature afterwards. The protocol's purpose is to make it virtually impossible to fake identification documents. Figure 2.7 shows the general procedure of Chip Authentication.

Chapter 3

Theoretical Considerations

This chapter focuses on the question as to which data regarding the considered identification documents is available and should be collected for an in-depth analysis as well as the possible attacks that can be carried out if this analysis reveals a weakness.

3.1 Use Cases

Within the following paragraphs, the use cases of the German Identification Card are described.

3.1.1 General Procedure

Independent of the concrete case, the general procedure of a data exchange with the Identification Card is the same. At first, the reader device performs the Card Select Procedure to set up a communication channel. Then, PACE is executed to provide encryption for the channel. The protocol takes place only between the reader device and the card. As a last preparative measure, Terminal and Chip Authentication are performed between the card and the device that actually tries to access the data. This is not necessarily the reader

itself, but often a computer connected to the reader or a server on the internet.

3.1.2 Governmental Versus Civilian Usage

If the card is to be accessed by governmental authorities, the MRZ or the CAN are used as passwords for the PACE protocol. This enables the accessing party to read data from the chip without the cooperation of the owner and without the need for the PIN which can be forgotten. Governmental authorities have certificates for the Terminal Authentication protocol that allow them to access all data on the chip, such as the biometric image, fingerprints (if they are stored) and the address of the owner.

On the contrary, when a civilian party wants to query data from the Identification Card, the password for the PACE protocol is the PIN. This prevents misuse as the cooperation of the owner is required for every interaction. The certificates for Terminal Authentication usually allow only certain information to be read, such as the name and address for online shops. There is support for an age verification on the chips, that allow the service provider to verify that the owner is above a certain age without giving him access to the date of birth itself. A similar verification is available for the city of birth. Furthermore, the provider can perform a *Restricted Identification* to recognize returning customers. The protocol offers an identifier that is constant for a single card, but is different from one service provider to another, to prevent the creation of behaviour profiles by cooperation.

3.1.3 Online Versus Offline Usage

The most simple use case is that the card is placed on a reader that either processes the data itself or is directly connected to a computer which handles the processing. This, for example, is the case at border controls.

There are two possible attack scenarios for offline usage. On the one hand, the attacker can place a sniffer near the reader and intercept all communication conducted with the card. Of course, the channel is encrypted after the execution of PACE, but the attacker can still log raw packets and measure timings for later use. On the other hand, the accessing party itself can be

the attacker. For governmental authorities this is rather unrealistic, as they already have access to all data. For civilian service providers however it can be interesting to try and access data that they are actually not allowed to. Since they already provide the reader and the processing device (which might be the reader itself), they have access to all data that is sent by the RFID chip on the card as well as the timing information if they measure it.

The protocols of the Identification Card were designed to make an online usage possible as well. The reader is supplied by the card owner and is connected to a computer. The Card Select as well as PACE are executed between card and reader only. Terminal Authentication, Chip Authentication and all following communication takes place between the card and a server which is supplied by the service provider.

Similar attack scenarios as for the offline usage are possible online, although the risk of an attacker placing a sniffer next to the owner's card reader is significantly smaller. If the attacker is the service provider himself, he can however not access the same amount of information as in the offline scenario. Since no information reaches the server until after the execution of PACE, he can thus only log the traffic of Terminal Authentication, Chip Authentication and the data exchanged thereafter. Timing information is available, but most likely not of much use, as the internet introduces a high fluctuation of latency.

In addition to this, an attacker can try to mount a man-in-the-middle attack on the connection to the server. The data stream is still encrypted by Terminal and Chip Authentication (and possibly an encrypted connection between the owner's computer and the server, such as SSL). If he is unable to crack this encryption (which is very likely due to the protocols in use), he can only log ecrypted data. The timings he can gather are also subject to the internet's fluctuation and are thus most likely of little use.

3.1.4 Hidden Readers

While it is not a use case per se, an attacker can place a hidden reader at arbitrary locations to gather information from Identification Cards that pass its vicinity. Due to the fact that the reader is not supplied by the card owner and that the communication does not go beyond the reader, this case will be grouped with the offline scenarios.

Unless the adversary is an institution entitled to access data on the cards for a legitimate use, for example a shop owner, he will not have a certificate for Terminal Authentication. It is also very unlikely that he has knowledge of any MRZs, CANs or PINs of the cards that pass the reader's field. Thus, he can merely read basic data such as the ISO 14443 UID or information about the supported encryption algorithms and gather the timings during this procedure.

3.2 Card Type Distinction

The protocols and standards related to the German Identification Card often do not dictate certain behaviour for implementations but rather give recommendations and choices for the actual manufacturers to decide during implementation of the card. Since there are two different companies manufacturing the chips for the Identification Cards [37, 17] and the companies might change their implementations from time to time while still conforming to the standards, there might be characteristics that allow an adversary to distinguish between a limited number of different cards.

3.2.1 ISO-14443 Type A/B

The ISO-14443 standard [24, 25, 26] allows two different kinds of RFID cards that use different modulation schemes in the physical communication layer as well as a different command set in the card selection protocol. These are called type A and type B cards. A reader that is to be certified as being ISO-14443 compatible must be able to communicate with both types of cards. The new German Identification Card is not bound to one type, so it might accomodate a chip that supports either of the two types or (possibly, but unlikely due to increased manufacturing costs) both.

Since the reader device has to use different commands for the communication with type A and type B cards, it can easily detect the type of a specific card. This information allows an adversary to divide all recognized cards into two sets, offering 1 bit of entropy.

3.2.2 Distinction via Protocol and Algorithm Support

For the implementation of the PACE protocol according to its definition [19], a number of parameters has to be set. These parameters include the protocol version, whether to use elliptic curve cryptography, encryption algorithms or the key length. A lower bound for the number of theoretically possible distinct parameter sets is 32, but this number is much higher in reality as there is no fixed number of available domain parameters that are used during the key generation. Together with all other parameters that have to be specified on the German Identification Card (e.g. for Terminal and Chip Authentication), this results in a lower bound of 394240 distinct sets. The composition of all this information could make up a "fingerprint" of a certain card and allow a recognition in a larger set of similar cards.

However, the values for most parameters have been fixed by the BSI [8], so it is not possible to distinguish cards based on such a fingerprint. The reason why the choice of parameter values has been left open in the first place is to give other institutions who want to include the same protocols in their products the chance to adapt the parameters to their needs.

Instead, it is possible to read the list of supported algorithms and parameters from the card and check the ordering of this list. There is no regulation regarding the ordering, so it might provide a degree of entropy that can be combined with other features to allow the identification of a card among a set. 6 parameter entries have to be included on every chip, so the lower bound for the number of possible orderings is 720. It is possible that this is higher in practice, because the choice whether to include other parameter values in the list is left to the manufacturers.

3.2.3 Chip Authentication Public Key

In the Chip Authentication protocol (see section 2.2.4), the Identification Card uses a static Diffie–Hellman key to set up a secure communication channel between itself and the reader. The guidelines for the new German Identification Card [8] state that every card within one generation should have the same static key. In practice, the time interval of one generation is 3 months.

The standard requires readers to go through the Terminal Authentication procedure (see section 2.2.3) before initiating Chip Authentication. If an attacker manages to gain access to the public key without using Terminal Authentication, he could use this property to further divide cards up into smaller sets, possibly enabling him to recognize a previously seen card among a set of others.

3.2.4 Possible Attack Scenarios

Since Identification Cards are produced in large numbers by only two manufacturers, it is near impossible that one card can be distinguished from all other cards currently in use by examining the aforementioned properties. However, it is entirely possible that the combination of properties permits a distinction from a large set of other cards. The average size of such a set where each card shows distinctive properties depends on the entropy of these properties which can be approximated by sampling data from a large number of Identification Cards.

The data can—with the exception of the ISO 14443 card type, which is only available to the card reader or a sniffer—be accessed in both online and offline scenarios. All information passes the reader, goes on to the processing device and is unencrypted unless it is sent over a secure connection set up for this.

A scenario where this information could be used by an adversary to track individuals is for example a shop with a customer base that is fixed to a certain degree and very small, like an expensive fashion label. Even though the owner of the retail chain may not be authorized to read the Identification Cards of his customers, the installation of devices that secretly gather information could enable him to build movement profiles of the people coming to the shops by tracking digital fingerprints established from the data mentioned above.

3.3 Random Numbers

The protocols implemented on the new German Identification cards employ random numbers at various points. If the random number generator(s) used by the cards have flaws, the generated numbers might be predictable, allowing

an attacker to recognize a chip he has seen before by analyzing the random numbers of followup communication.

The BSI has specified requirements regarding the random number generators to be used on the Identification Cards [8], but these requirements do not apply for the ISO 14443 UIDs. Also, related analysis of random numbers in cryptography [5] has shown that in practice weak implementations can be found in places where they form an essential part of security measures.

3.3.1 Card Selection UID/PUPI

In the ISO-14443 Card Select protocol [25] (see section 2.2.1), the Identification Card generates a new UID/PUPI every time it enters the field of a reader. By cycling the field and probing the card multiple times in a row, the generated UIDs/PUPIs can be collected and analyzed for weaknesses in the employed random number generator.

3.3.2 PACE Random Nonce

In the initial step of the PACE protocol (see section 2.2.2), the chip generates a random number, encrypts it with a password (either the CAN or a key derived from the MRZ) and sends it to the reader. This number is called a *nonce* (**n**umber used **once**). Since the encryption key is static, a weak random number generator could allow an attacker to predict the packet sent by the chip even if he has no knowledge of the used password.

The PACE protocol can be initiated immediately after the ISO-14443 Card Select protocol without having to authenticate the reader to the card [19], because after its execution no data is made available that is not already visible on the Identification Card itself. When the packet has been received, the field powering the chip can be cut and the procedure can be repeated an arbitrary number of times until sufficiently many numbers have been collected.

3.3.3 Terminal Authentication Challenge

During Terminal Authentication, the card sends random data to the reader which is to be signed by the latter. The standard requires the PACE protocol to be executed before starting Terminal Authentication, but an implementation of PACE would be beyond the scope of this work. If the random data used during Terminal Authentication can be requested without a prior invocation of PACE, it would provide an additional source of random numbers that can be analyzed.

3.3.4 Possible Attack Scenarios

While the UID and the PACE nonce are only available at the reader (and can be captured by a sniffer), the Terminal Authentication Challenge is sent to the processing device, whether this is a server on the internet or the reader itself. The challenge is however encrypted using an ephemeral PACE key, so unless the attacker is able to crack PACE, it can only be read (and possibly exploited) by the service provider.

The predictablity of random numbers can lead to a very high amount of entropy available for recognizing previously seen cards. If, for example, the 24 supposedly random bits of the UID and 16 bytes of the PACE nonce could be predicted with certainty, one could recognize a previously seen card among a set of 1 million with a confidence of almost 100%. This is given by a generalization of the birthday problem [13].

An attacker could, for example, stand near his victim and retrieve numbers from the victim's Identification Card by using a hidden reader. Once enough numbers have been gathered, the next number could be computed and a large network of hidden readers that are positioned e.g. in shopping centers or pedestrian precincts could search for a card that generates this number when queried. If such a card is found, it can be assumed with high confidence that this card belongs to the victim.

A problem with this attack might arise if the number can only be predicted approximately and the victim uses his card at a different reader device in the meantime, as this would lead to the predicted number being "consumed" in the

process. The next numbers could not be predicted reliably and a new number of samples would have to be collected from this victim.

3.4 Timings

Chotia and Smirnov [11] successfully traced electronic passports by replaying BAC protocol [21] packets and measuring the times between the sending of the packet and the reception of an answer. It was discovered that packets captured from a previous communication with the same passport showed different timing characteristics than packets from another one.

The reason behind this weakness was that packets captured from a transmission by the same passport have a correct checksum, but an incorrect content. Messages from a different passport have an incorrect checksum as well as an incorrect content. The chip on the passport verifies the checksum first and if this fails, immediately returns an error message to the reader instead of decrypting the message and verifying the content. This difference could be measured and the passport could be identified with almost no doubt.

The BAC protocol is not implemented on the new German Identification Card [8], but weaknesses that allow for a similar exploitation might appear in the replacement PACE (see section 2.2.2). In particular, data concerning the domain parameters called "Mapping Data" in the standard [19] is exchanged after the encrypted nonce and before the public keys (between steps 3 and 4 in figure 2.5). Processing this data might lead to errors on the RFID chip at different points in time, depending on whether the replayed data was captured from a communication with the same or a different card.

3.4.1 Possible Attack Scenarios

As already mentioned in the first part of this chapter, timings are most likely useful only in offline scenarios or when the attacker is able to place a sniffer near the reader.

The PACE protocol can be executed by using one of 3 possible passwords: The MRZ, the CAN and the owner's PIN (the MRZ and the CAN are only

used by governmental authorities). If the timings observed when replaying messages allowed for a recognition of a previously seen card, an adversary could eavesdrop on a communcation with a legitimate reader device—e.g. by standing next to the victim at a border control or in a shop. Once he had a set of messages, he could recognize the victim's Identification Card within a large set of others.

If the intercepted PACE execution used the PIN or CAN, he would be able to identify cards that had the same PIN or CAN. He could not be absolutely certain, though, that the recognized card was a different one that only had the same password by chance. If, however, a handshake using the MRZ as a password could be observed, this would allow the attacker to precisely identify the same card among an arbitrarily large set of numbers, because the MRZ (i.e. the card's serial number which is part of the MRZ and used to derive the password) is unique on each card.

3.5 Conclusion of Considerations

The classification of cards based on distinctive properties (as explained in chapter 3.2) appears to be little promising for the purpose of tracking cards. Only a small number of properties allow for a division of cards into subsets and these properties provide little entropy, e.g. 1 bit in the case of the ISO 14443 card type distinction. The ordering of algorithm and version identifiers on the cards seems to be the property which potentially carries the most entropy, but a short test with four different, randomly chosen Identification Cards indicated that the ordering is either fixed or there are only very few different orderings being used in practice—all four cards showed the same sequence.

While the protocol versions are fixed, some parameters can still be varied by manufacturers and issuers, so there is some deviation between cards. Reading the parameters of the four different cards showed that two at a time were using exactly the same parameters for all algorithms. It can thus be assumed that there is some, but not much entropy in this information—probably in the order of 1–6 bits. Due to this small distinctness of cards, the classification will not be further pursued in this thesis.

The three sources of random numbers mentioned above will be used to gather

sample data if possible. If the Terminal Authentication Challenge can not be requested without a proper execution of PACE, it will be left out. The gathered random numbers will be analyzed by a series of tests that have proved to be sufficiently exhaustive, e.g. the Die Harder test suite [7] or the method used by Zalewski [38]. It will be checked if the random number generators show weaknesses in the form of predictability which allows tracking of cards.

For the PACE protocol, timings will be measured and analyzed for anomalies that might allow recognition of a card. The protocol implementation will be analyzed to find out wether a similar flaw like the one discovered in the electronic passports [11] exists.

Chapter 4

Implementation

This chapter covers the details of the implementation used to obtain data for analysis. At first, the design of the Proxmark III toolkit used here is explained and the information flow within the toolkit is detailed. Thereafter the modifications, improvements and additions which were done to the existing Proxmark III software are listed together with the design ideas behind them. In addition to this, the method used to collect data for the in-depth analysis in the following chapters is specified.

4.1 Proxmark III Architecture

As outlined, the implementation took place on the Proxmark III RFID toolkit.

The Proxmark III toolkit consists of the client application running on a PC and 2 programmable hardware components—a Xilinx Spartan-II FPGA and an ARM7 microprocessor.

The client application is a command line utility written in C which enables the user to pick from the available abstract tasks such as ISO 14443 Card Selection or the interception of a communication between other devices. Commands are sent to the microprocessor via USB.

The microprocessor breaks down the task into its individual steps such as querying RFID cards for their UIDs or sending an APDU to one of them. It composes the packets and encodes them up to the link layer. They are then passed to the FPGA as a sequence of bits.

The FPGA is used as a signal processor to modulate the data which is to be sent and to demodulate the received signal.

Received data is passed to the microprocessor which acts accordingly. Once it has finished the requested task, the collected information is transferred to the client application which can then display it to the user.

Figure 4.1 shows the data flow between the components.

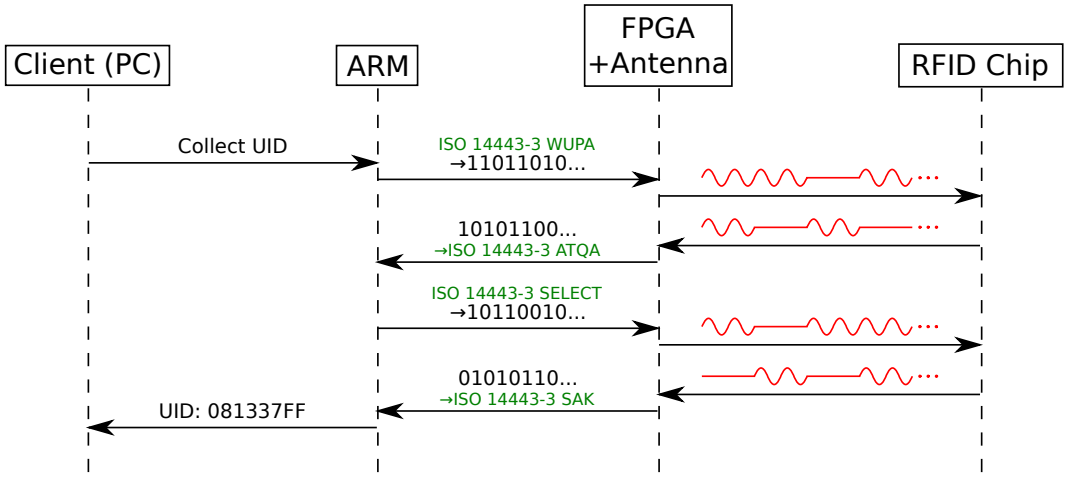


Figure 4.1: *The abstraction layers and the data flow of the Proxmark III toolkit. The encoding and decoding of ISO 14443 commands to and from bit sequences takes place on the ARM microprocessor.*

Before this thesis' source code was included, the software had offered basic ISO 14443 link layer functionality—Card Selection as well as APDU transmission and reception. Higher level protocols such as those required for the German Identification Card had not yet been implemented and the APDU functionality had been incomplete.

4.1.1 Client Usage

The Proxmark III command line client `proxmark3` is composed of command subtrees. Commands of similar nature are grouped under tree nodes that distinguish them from different commands, i.e. the top level nodes in the tree are used to distinguish between commands related to low-frequency RFID tags (`lf`), high-frequency RFID tags (`hf`) and others. By concatenating the tree nodes, the user specifies the context of the command to be executed. The command `hf 14a cuids` for example works with high-frequency (`hf`) RFID tags that are ISO-14443 conformant (`14a`) and collects UIDs (`cuids`).

If there is additional input separated from the command by a space, this is taken as a parameter to the command.

The commands related to this thesis are found under the `hf 14a` and `hf epa` subtrees. Figure 4.2 shows the relevant parts of the user interface. It also illustrates which functionality is included in which source code files for both the user interface and the microcontroller firmware. In the microcontroller code, the functions for the handling of the German Identification Card (in `epa.c`) make use of generic ISO 14443 functionality which is located in the file `iso14443a.c`.

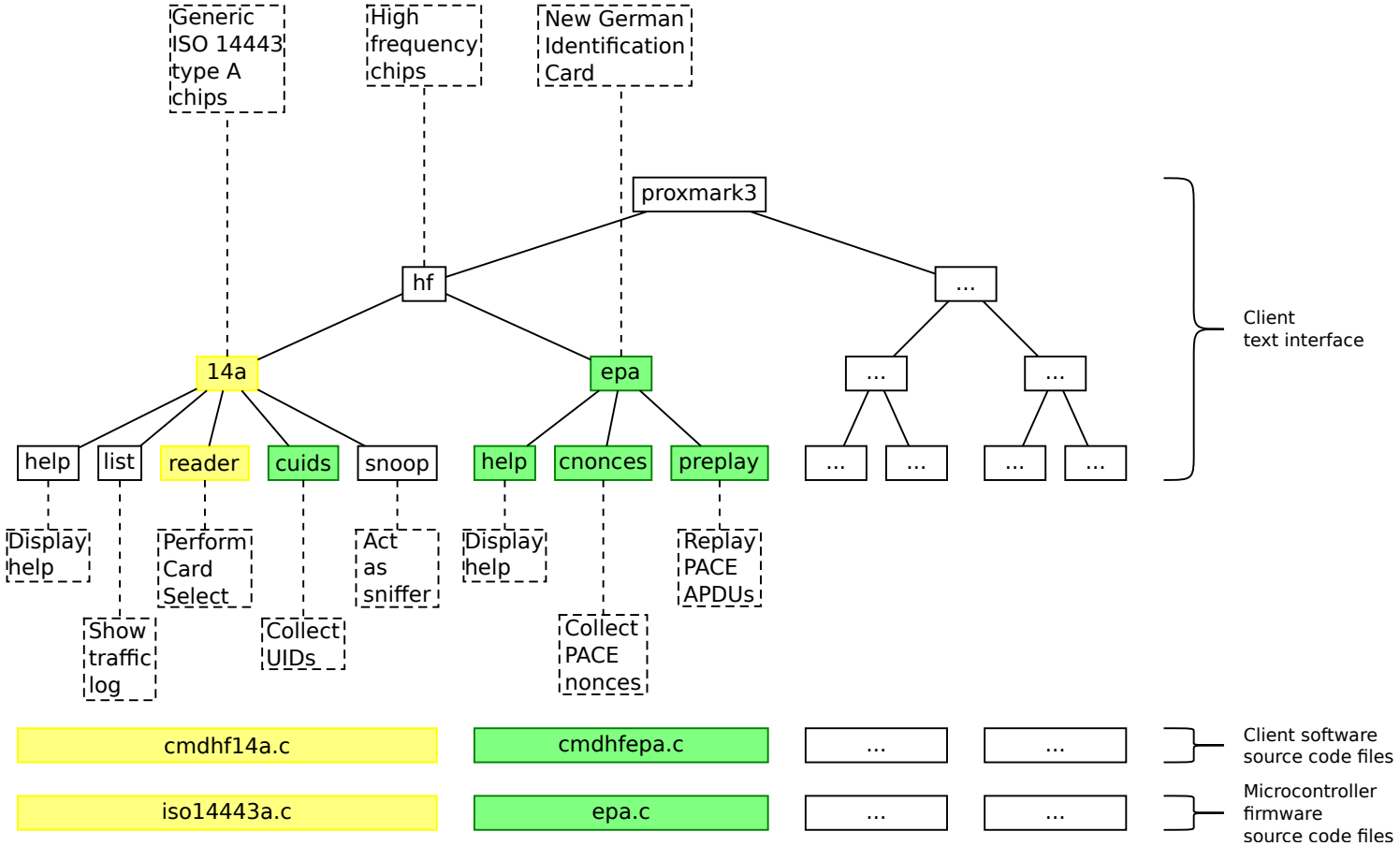


Figure 4.2: *The software architecture and user interface layout of the Proxmark III software. Parts marked in yellow have been updated to fix bugs or remove limitations. Parts marked in green have been added newly.*

4.2 Limitations of the Implementation

Before the implementation of the functions used in this thesis was complete, the Proxmark III software had only been capable of communicating with ISO 14443 Type A RFID chips (see section 3.2.1). There are no regulations for the German Identification Card as to whether it has to support the ISO 14443 Type A or Type B standard, so cards conforming to either of the two are existing and being used. However, the higher level protocols are not affected by this and there is little reason to assume that differences in the link layer implementation affect the distribution of random numbers from the generator. Thus, the software written to collect the data for this thesis supports cards of ISO 14443 Type A only at the time of writing.

4.3 General Modifications to the Proxmark III Software

As mentioned above, the functionality of sending and receiving APDUs had been incomplete in the original Proxmark III software. The ISO-14443 standard [26] states that the reader has to keep track of a bit which is included in every command and every response. Each time a response is received and the bit in the packet's frame matches the tracked bit, it has to be flipped to indicate that the next data transmitted to the card actually is a new packet.

Within the scope of this thesis the missing functionality has been implemented so that the APDU communication works reliably and is conformant to the standard.

According to ISO 14443-3 [25], each byte in the data stream which is transmitted from the reader to the card is to be followed by a parity bit. The original implementation managed the computation and inclusion of this bit, however, implementation details prevented packets larger than 32 bytes from being encoded correctly.

In order to support those parts of the protocols that were used to collect relevant data for this thesis, this restriction has been removed so that the maximum length of messages now only depends on internal buffer sizes that

can be easily increased when needed.

The new commands which were added during the course of this work were documented in the Proxmark III wiki [35] on the project's website.

4.4 UID Collection

Before the start of this thesis, the existing firmware for the Proxmark III already supported for the ISO 14443 Type A Card Select procedure. The card is queried and the UID together with some additional data is passed to the client. Figure 4.3 shows this procedure.

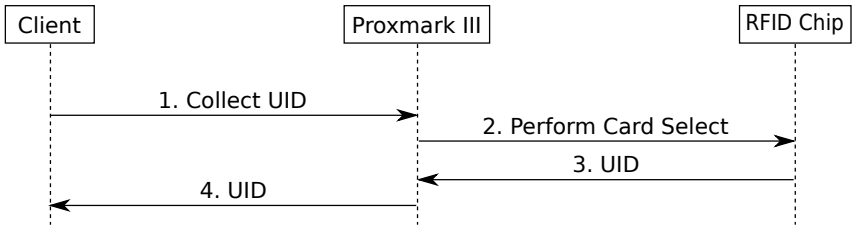


Figure 4.3: *Collection of UIDs using the Proxmark III. In order to query a large number of UIDs from the same card, this procedure is repeated an arbitrary number of times by a loop in the client software.*

By implementing a function in the client which sends the same command to the microprocessor multiple times, it has been made possible to collect a batch of card UIDs in a single run without user interaction. The function has been called `cuids` and has been made available under the `hf 14a` command subtree. It takes the number of UIDs to collect as a single argument and prints them, together with a timestamp from before and after the operation. This allows for easy collection and further processing by scripts and programs.

4.5 PACE Nonce Collection

In order to collect the nonces generated by the card during PACE, the existing basic ISO 14443 functionality was used to execute the initial part of the protocol. The procedure is illustrated in figure 4.4.

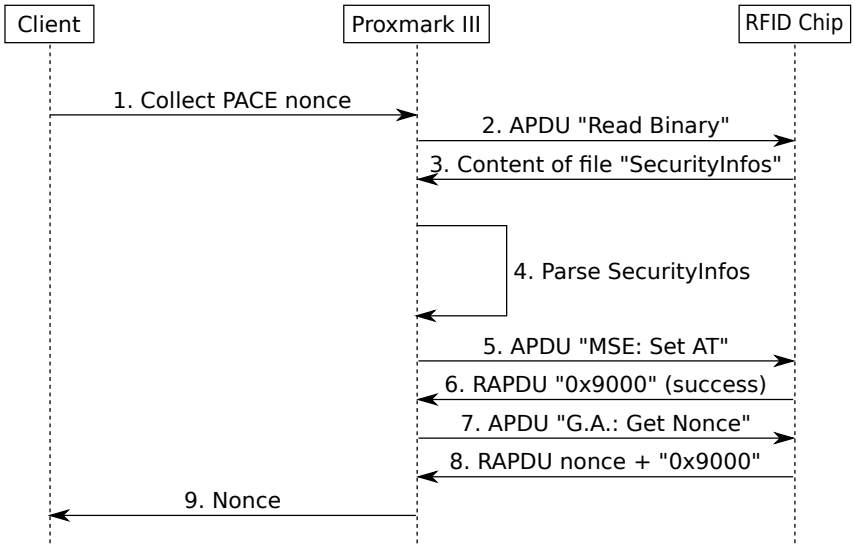


Figure 4.4: *Collection of PACE nonces using the Proxmark III. Similar to the UID collection, a loop on the client side facilitates the sampling of larger numbers of nonces.*

The software has been extended to perform the first part of a PACE handshake after the ISO 14443 Type A Card Select procedure. It selects the card and then queries it for supported PACE protocol version and algorithm identifiers. It then uses this information to initialize the PACE protocol by sending an `MSE: Set AT` APDU [19] to the card. Once the card indicates that the initialization succeeded, the reader requests an encrypted random nonce by issuing a `General Authenticate: Request Nonce` APDU [19]. After reception, the nonce is transferred to the client and the communication with the card is aborted.

In particular, this is done by a protocol handler that verifies the result of each performed step during the procedure. A parser was implemented to find the necessary information in the card's list of supported algorithms and parameters. At the time of writing, the parser only finds the values relevant for PACE, but it has been written in a way that allows easy extension and reusability. If the need arises, the results of the implementation can be used to extend the Proxmark III software so that it supports the PACE protocol entirely.

The implementation of the nonce collection initiates PACE with the CAN as the password. This prevents the card from locking up as would be the case if the PIN was used.

4.6 Terminal Authentication Nonce Collection

To check whether the card would allow the initialization of the Terminal Authentication Protocol before successfully completing a PACE handshake, an *MSE: Set AT* APDU was issued by making use of the existing functionality. The test showed that the card responds with an error code as expected, thus no further investigation has been done on this matter.

If the Terminal Authentication Protocol had been accessible, it would have provided another source of random numbers for analysis. It is however unlikely that there are 3 different implementations of random number generators on the Identification Card. Thus, it is assumed that the nonce generator for the PACE protocol also handles the generation of nonces for the Terminal Authentication protocol.

4.7 PACE Timings

By using the capabilities of the Proxmark III software that had already existed together with the implementation of the PACE nonce collection, the firmware and client have been extended to allow replaying of a captured PACE handshake.

A protocol handler similar to that of the PACE Nonce Collection has been established. In contrast to the latter, it does not use a parser but merely sends saved command APDUs and handles the reception of response APDUs. Figure 4.5 shows a measurement using the implemented function.

5 APDUs in hexadecimal notation—e.g. `10860000027C0000` as the `General Authenticate: Get Nonce` APDU—need to be entered into the Proxmark III client software. They can be taken from an intercepted communication with the same or a different card. These APDUs are then transferred to the reader and are used to perform each step of the PACE protocol. The Proxmark III replays one APDU at a time, verifies the status word included in the card's answer and continues with the next APDU.

While doing so, the reader measures the times it takes the reader to send and the card to respond to each of the 5 APDUs used during the protocol execution

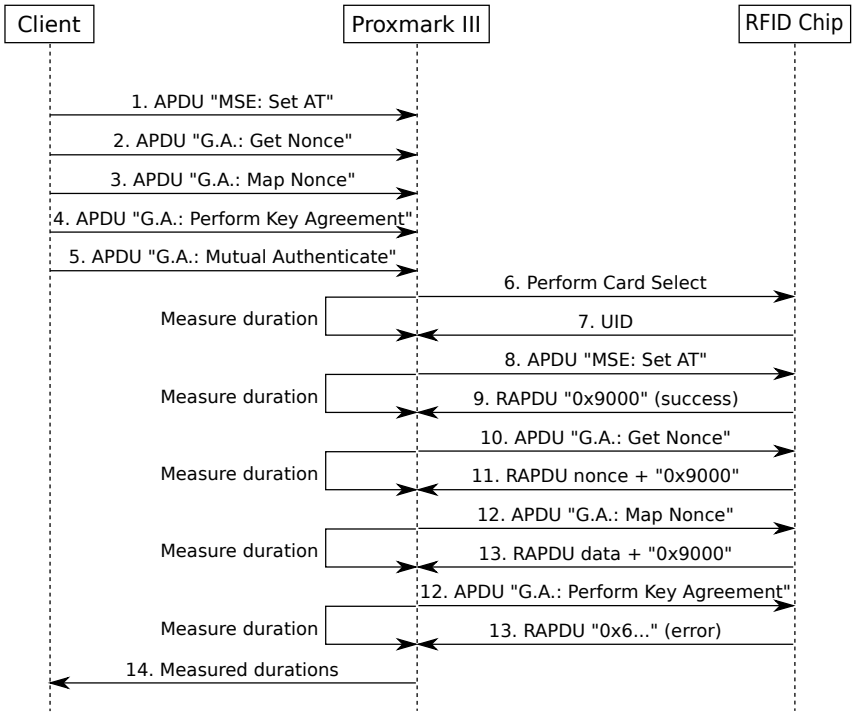


Figure 4.5: *A replay of PACE APDUs using the Proxmark III. Although the chip usually responds with an error code of* `0x6985` *or* `0x6A80` *after the* `General Authenticate: Perform Key Agreement` *APDU, the* `Mutual Authenticate` *APDU is still saved on the Proxmark III for the unexpected case where the chip accepts the replayed data without an error. In that case the last APDU is also sent and the duration is measured as well.*

with a precision of $1\mu s$. Upon completion, the times are passed back to the client where they are printed for the user. If the card responds with a different status word than `0x9000` (successful completion), the process is aborted and the times measured up to this point are passed back and printed.

Because the Proxmark III software is not yet capable of performing a full PACE handshake on its own, the APDUs that should be replayed need to be collected using other soft- and hardware, e.g. by logging the communication of the AusweisApp [3] or other software [20] and using a PC/SC-compatible reader device.

Chapter 5

Analysis Methods

In this chapter, the methods used to analyze collected data are described in detail. The random numbers are analyzed using two different approaches and the timings of a PACE protocol execution using replayed messages are given a closer look as well.

5.1 Random Number Analysis

The analysis of random numbers and random number generators is a complex field where much work has been done in the past [30, 31, 7, 38]. Starting with a naive approach, a random number generator is expected to produce numbers within a certain interval with a uniform distribution. However, this criterion alone is far from enough to ensure that the numbers are actually unpredictable. The difference of subsequent numbers and the distribution of single bit values for example are other aspects that can provide an adversary with subtle information about the future values being given by the generator.

Two sources of random numbers have been found to be accessible on the German Identification Card. They have been used to collect sample data that allows an analysis of the random number generators implemented on the card. For the analysis, two methods were used that have been shown to detect flaws in existing random number generators in the past [38, 6]. These two methods

are described below.

5.1.1 3-dimensional Attractors

In 2001, Zalewski used 3-dimensional attractors to detect flaws in random number generators that provided initial sequence numbers for the TCP protocol [38].

An attractor can be defined as the evolution of a dynamic variable a over time t, expressed as $a(t)$ (this is not nearly as precise as the definition of attractors in mathematical literature [32] but suffices for its usage in this thesis). In this case, the dynamic variable is the last generated number of a random number generator.

At first, a sample set of random numbers is collected from the target generator. This set ideally contains all possible numbers produced by the generator, but in practice a set of 50,000 numbers has proven to suffice when dealing with 32-bit integers.

From the set an attractor is built by using a method called *delayed coordinates*. From the 1-dimensional sequence of input numbers $A = \{a(1); a(2); a(3); \ldots; a(|A|)\}$, 3-dimensional coordinates are calculated by defining

$$\begin{aligned} a_x(i) &= a(i) - a(i-1) \\ a_y(i) &= a(i-1) - a(i-2) \\ a_z(i) &= a(i-2) - a(i-3) \\ \forall i &\in [4; |A|] \cap \mathbb{N} \end{aligned} \tag{5.1}$$

as the x, y and z coordinates, respectively.

The so gained set of points in 3-dimensional space provides the mentioned attractor which is specific to the given generator.

For the test itself, 3 sequential random numbers are queried from the generator. These can be used to construct the y and z coordinates of a line in 3-dimensional space using formula 5.1. The integer x coordinates of all points on this line are the possible values for the next number given by the generator. Zalewski states “a widely accepted observation about attractors”, namely that “if a sequence [of numbers from a random number generator] exhibits strong

attractor behavior, then future values in the sequence will be close to the values used to construct previous points in the attractor" [38]. This means that a weakness in the generator can lead to numbers being predictable by taking points on the previously constructed line that are in or near the set of points computed from the sample data. Figure 5.1 shows an example for this case.

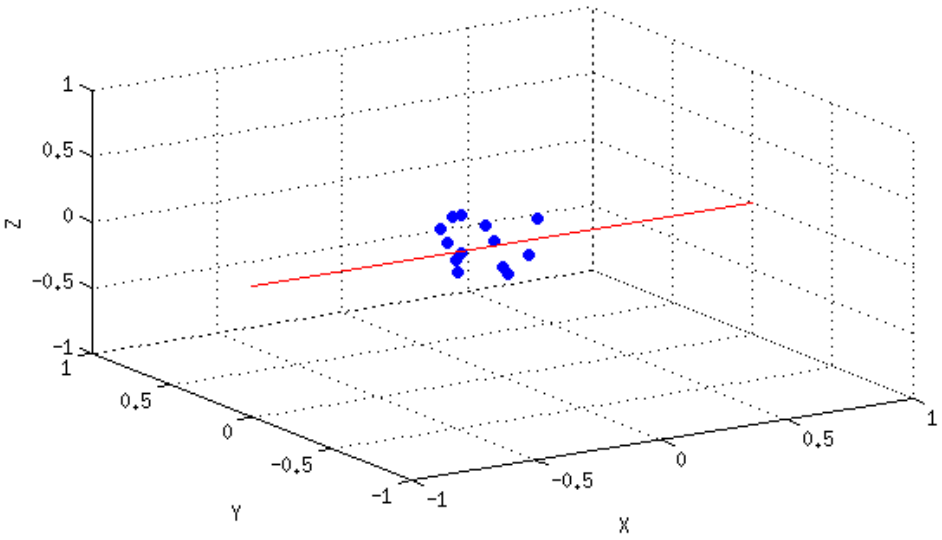


Figure 5.1: *An example of a 3d-attractor-based analysis of a random number generator. The red line has been constructed by querying 2 numbers from the generator and computing the coordinates using formula 5.1. If the generator is flawed in a certain way, the next number (when combined with the 2 coordinates forming the line) will lead to a point close those made from the sample data (in blue). This means that it will be close to the previous value (the X coordinate will be close to 0).*

In the analysis of the random number generators used for the TCP protocol's initial sequence numbers, Zalewski took a set of 5000 numbers that were on the line and near the sample points. He called this set a *spoofing set* and used it to guess the next initial sequence number used by the host by creating one packet for each possible initial sequence number.

For the analysis of the random number generator implemented on the Identification Card, this technique can be used similarly. After taking sample data from an arbitrary card and constructing the 3-dimensional attractor, 4 numbers are queried from the same card. From the first 3 numbers, a line is constructed using formula 5.1 and leaving the X coordinate variable. If there are only few

points in the attractor shape that are relatively close to the constructed line and if the the fourth number is one of them, then the generator is flawed.

If this is true for the actual cards which are being used by the public, a tracking attack could be mounted by querying 3 ISO 14443 UIDs or 3 PACE nonces from a victim's card and constructing a line by using formula 5.1 again. Using this line, the next number to be returned by the generator could be predicted with a certain confidence. A network of readers that query all cards in their range continuously could look for one that returns the predicted number. This could then be assumed to be the card of the victim, again with a certain confidence (given by the generalized birthday problem [13]).

The reliability of the results gained by such an attack depends mainly on the shape of the attractor and the correlation between the actual results and the constructed attractor. If, for example, the attractor for the ISO 14443 UIDs is a 24-bit (2^{24}) wide cloud (in X direction), then the results will not be useful at all because it does not give a hint on subsequent numbers (24 of the 32 bits that form a UID are random). If, however, the size of the cloud or the size of clusters within the cloud is very limited in X direction, the chances of positively identifying a previously seen card are much higher.

5.1.2 Dieharder Random Number Test Suite

Dieharder [7] is a software suite created by Robert G. Brown and designed for testing random number generators to find possible weaknesses. It incorporates a total of 31 individual tests at the time of writing. 18 of these are reimplemented tests from the diehard software suite by Marsaglia [15], 3 are reimplemented tests from the NIST STS suite [36] and 10 are various individual tests by Brown and other persons. Of the 18 tests, 3 have been marked as being "suspect" regarding their reliability and 1 has been marked as being entirely unreliable. This has not been proven, but the remaining tests supposedly analyze enough different aspects of the generator to be sufficient.

The dieharder suite's analysis is based on the principle of contraposition [6]. It assumes that the tested random number generator is perfect. This assumption forms the *null hypothesis*. Based on this assumption, certain further properties of the numbers given by the generator are stated. As an example, sums over sequences of 1-bit integers from an ideal random number generator would have

a mean value of $\frac{1}{2}$ and a standard deviation of $\frac{1}{4}$. The program now takes a large number of such sums from the given random number generator and computes the probability (called *pvalue*) of obtaining such values from an ideal one. This is repeated a number of times and the resulting *distribution of pvalues* is tested for uniformity. If this test yields a probability for the distribution being uniform of less than a certain value (0.000001 by default), that means that the probability of this generator being an ideal one is very low. Thus, the null hypothesis is rejected. The exact calculation of the individual *pvalues* differs from test to test, but the overall test for uniformity stays the same.

Obviously, even an ideal random number generator would fail a test from the dieharder suite every now and then (for any finite number of test runs), but the parameters like the amount of numbers in each sequence and the *pvalue* threshold can be tuned to make this extremely unlikely. For the same reason, even a weak generator can pass these tests and still be predictable, but the number of different tests and again the customizable parameters make this similarly improbable.

Subsequently, 3 of the dieharder tests are explained in more detail to give a brief overview of the different aspects of analysis the suite tries to cover. The descriptions are aggregated from the dieharder program's output as well as the individual whitepapers or test descriptions [30, 29, 36, 18].

The Diehard Birthdays Test

The diehard birthdays test is a random number generator test designed by George Marsaglia for his diehard suite [30, 29]. It takes m integers in an interval $[1; n]$ from a generator. The numbers are ordered ascending and the

differences between each two consecutive values is computed:

$$\begin{aligned}&\text{Given}\\&1 \leq v_1 \leq v_2 \leq v_3 \leq \cdots \leq v_m \leq n\\\\&\text{Compute}\\&d_1 = v_1\\&d_2 = v_2 - v_1;\\&d_3 = v_3 - v_2;\\&d_4 = v_4 - v_3;\\&\ldots\\&d_m = v_m - v_{m-1}\end{aligned} \tag{5.2}$$

In this new sequence of numbers duplicates are counted and the number of values that appear more than once is defined as a new variable

$$Y = |\bigcup_{i=1}^{m}\{d_i : \exists j \neq i : d_i = d_j\}| \tag{5.3}$$

This random variable is assumed to be Poisson-distributed with $\lambda = \frac{m^3}{4n}$ for an ideal generator. Although this assumption is yet unproven, it is strongly supported by the fact that most supposedly "good" random number generators (that pass all other tests) do in fact exhibit this distribution when tested. Because of this knowledge, the actual distribution can be compared to a Poisson-distribution to obtain the likeliness of this generator being flawed.

The name of the tests comes from the fact that it is inspired by and partly based on the *problem of duplicate birthdays*, the probability of two persons having the same birthday in a given set of people.

The default parameters for this test in the dieharder suite are as follows. 512 24-bit integers are generated and the duplicate spacings are counted. This is repeated 100 times and the resulting distribution is analyzed to form 1 *pvalue*. 100 such *pvalues* are computed for the overall test. This nets in a consumption of 15.36 MiB of random numbers for the complete test.

The STS Serial Test

The purpose of the Serial Test from the NIST STS suite [36] is to find bit-level correlations in weak random number generators. For a given parameter m, it

counts the occurences of each bit pattern of length m in the bitstring given by the generator. Since the distribution of single bits should be uniform, the distribution of the bit patterns should be uniform as well.

The exact calculation however is more than mere counting, though. When counting the occurences in a non-overlapping manner—that is when dividing the input into chunks of m bits each—the results might be different than when counting in an overlapping manner. The bit sequence 11110001100110011010 1010011 for example will yield a count of 0 occurences of the bit pattern 000 when counting non-overlapping, but one can easily see that the pattern does occur in the input (5^{th} to 7^{th} bit). This might lead to false rejection of a good generator.

However, when counting overlapping bit-patterns, there is a significant correlation between subsequent patterns—if the first 3-bit pattern of a sequence is 000, the second one can only be 000 or 001, obviously. To account for this correlation in the calculated distribution, the test not only counts the occurences of all patterns of length m, but also of those of length $m-1$ and $m-2$. The 3 distributions are combined in an overall statistic which follows a χ^2 distribution with 2^m-1 degrees of freedom for an ideal random number generator. The exact formulae can be found in the test specification [36]. A special feature of this test is that it produces two *pvalues* per run instead of just one.

With the default parameters of dieharder, all sequences of length 1-16 are counted. For each length, 100000 samples are examined to generate one *pvalue* and 100 *pvalues* are used for the overall statistic. Thus, the complete test for all lengths consumes 170 MiB of random data from the generator, of which the longest test—16 bit sequences—consumes 20 MiB.

The Dieharder Generalized Minimum Distance Test

This test, based on two tests from the diehard suite [29], was developed by Fischler [18] and implemented by Brown for his dieharder suite [7].

The original tests by Marsaglia takes a certain number of random points from the generator within a 2D square or a 3D cube of fixed size by interpreting consecutive numbers as components of 2- or 3-dimensional coordinates. It then calculates the minimum distance between 2 such points. These minimum

distances taken to the power of 2 or 3 for the 2D or 3D tests, respectively, should then be approximately exponentially distributed. By measuring the actual distribution of the values from the tested generator and comparing it to the known exponential distribution, a *pvalue* can be received for each set of points.

Fischler proved that the exponential distribution used by Marsaglia was not quite that of an ideal random number generator [18]. A correction had to be included in the distribution depending on the number of dimensions used. The revised distribution includes a calculated part of this correction. It does not include the theoretically accurate correction, but is precise enough for practical application not to give false results. Fischler also calculated the corrections needed for up to 5 dimensions and showed what needs to be done to apply the test for even more dimensions.

Brown used the results of Fischler's work to implement a test for his dieharder suite that applies the test for 2-5 dimensions [7]. The test uses the 32 bits to specify one dimension of a coordinate. Each generated point thus consumes $4*d$ bytes of random data, where d is the number of dimensions used. For each number of dimensions from 2 to 5, 1000 *pvalues* are calculated and for each of the *pvalues* 10000 points are generated. This means that for the complete test, 560 MiB of random data is consumed, the 5-dimensional test alone consumes 200 MiB. As with most other tests, these parameters can be varied, though a number of dimensions lower than 2 or higher than 5 is not yet supported due to the statistical corrections being unknown.

5.2 Timing Analysis

The analysis of timings in the PACE protocol is fairly straightforward. First, a successful authentication using the PACE protocol is executed by using the GlobalTester software [20] which supports the protocol and is able to create log files including all data exchanged between the card and the reader. This is repeated for a set of available cards, the command APDUs sent from the reader device to the cards is extracted from the log files and saved.

Subsequently, the Proxmark III is used to replay the gathered APDUs to two cards. It is analyzed as to whether the time it takes the card to respond to

each APDU differs between the sets of recorded packets. To have an approximate measure of the deviation in timings, APDUs of each set are replayed approximately 100 times and the timings within a set are accumulated. The result should then show whether there is a reliably identifiable difference in timings between the two traffic patterns.

Chapter 6

Analysis Results

This chapter focuses on the results gained by analysing the collected data using the methods described previously.

For the analysis, a total of 464151 ISO 14443 UIDs and 278991 encrypted PACE nonces have been collected from a sample Identification Card that is not in productive use due to missing long-time availability of other cards. If the results do not show a weakness in the implementation, it is thus assumed that the Identification Cards used in public do not suffer from weaknesses that had not already appeared on this sample card.

The collection of a single UID using the Proxmark III toolkit takes approximately 0.44 seconds and the collection of an encrypted PACE nonce takes approximately 1.02 seconds, resulting in net data rates of approximately $6.81\frac{B}{s}$ and $15.69\frac{B}{s}$, respectively. Due to this, the amount of sample data available for testing has been limited and some tests—i.e. from the dieharder suite—might not provide sufficiently meaningful results. This is mentioned in the respective explanations and further testing with larger data sets is recommended.

Timing analysis has been conducted with different cards of which most are Identification Cards owned by existing persons and are actually in use. These cards have been chosen randomly and are not known to be correlated in any way.

6.1 3-dimensional Attractors

The 2 accessible random number sources have been subjected to attractor analysis as explained in chapter 5.1.1. The interpretation of the obtained results is presented subsequently.

6.1.1 ISO 14443 UIDs

The collected ISO 14443 UIDs have been plotted into a 3-dimensional graph using formula 5.1.

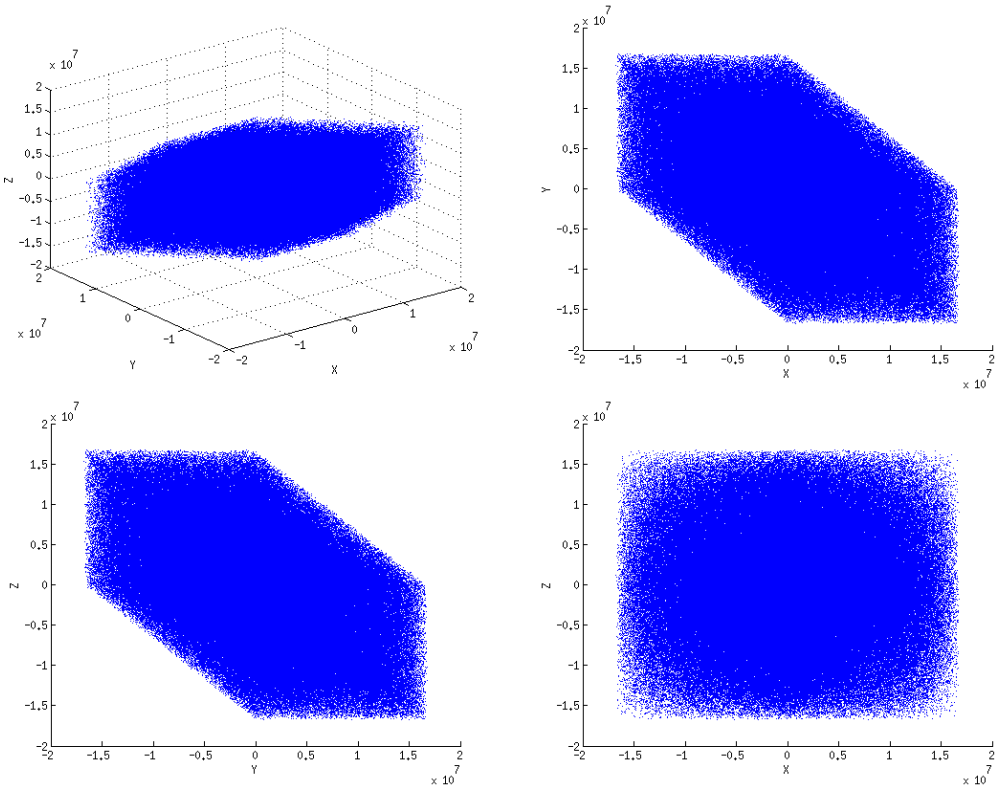


Figure 6.1: *The resulting graph for the attractor of the ISO 14443 UID generator. As one can immediately see, the construct is a parallelepiped that has a width of* 2^{24} *in X direction at every point.*

Figure 6.1 shows the resulting attractor. Because of the parallelepiped having

a width of 2^{24} in X direction at every point, one can make no prediction of future UID values by using the technique of Zalewski [38]. No matter where the line created by combining two received UIDs is located, there are 2^{24} possible values for the next UID. The random number generator used in the German Identification Card thus does not experience attractor behaviour and can be considered sufficiently strong regarding this aspect.

6.1.2 PACE Nonces

For this analysis, the encrypted nonces from the PACE protocol have been, similarly to the UIDs, plotted to show the attractor behaviour of the underlying random number generator.

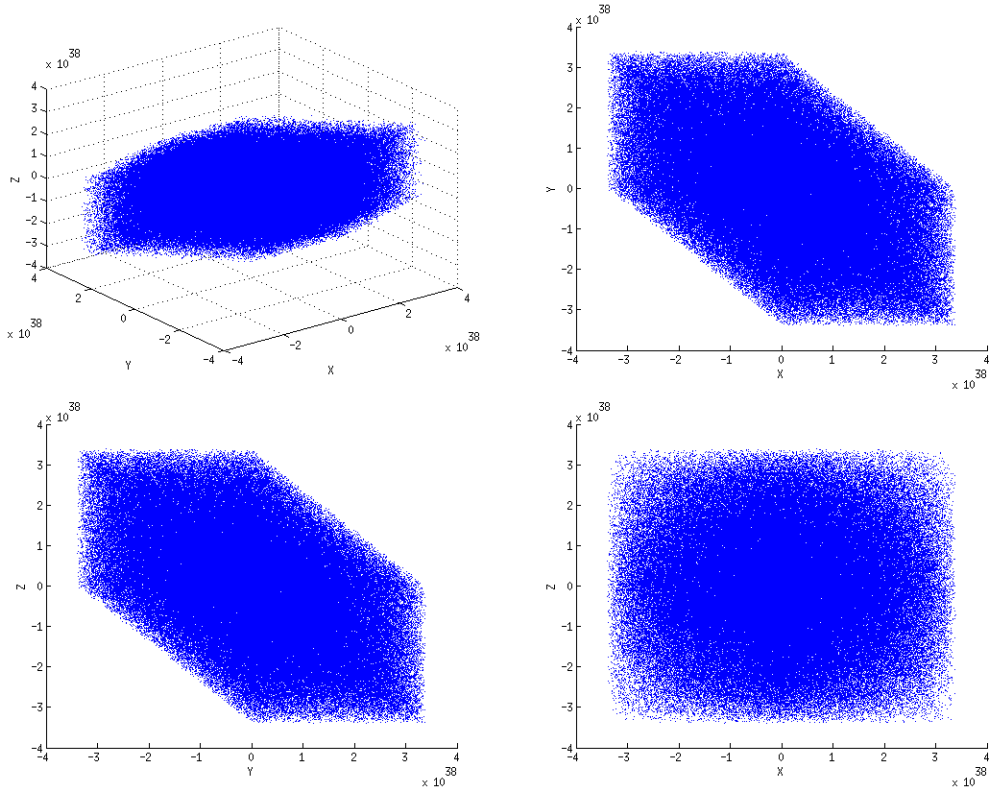


Figure 6.2: *The attractor figure for the encrypted PACE nonces. The graph looks similar to that of the ISO 14443 UIDs except that the width in X direction at every point is* 2^{65536}.

Figure 6.2 shows the resulting figure. As with the ISO 14443 UIDs, the attractor shape is a parallelepiped and thus no prediction can be done on subsequent PACE nonces. This generator, too, can be considered sufficiently strong regarding 3-dimensional attractor behaviour.

6.2 Dieharder Random Number Analysis

The collected random numbers have been analyzed using the dieharder random number test suite detailed in chapter 5.1.2. The results of the analysis and considerations that have to be made based on these results are shown here.

The default dieharder parameters have been used for the tests initially, but have been adapted to fit the amount of available sample data where possible if the tests indicated anomalies in the input. The results of 4 tests not included in the following paragraphs have been disregarded due to the fact that Browning, the author of the test suite, classifies them as being "suspect" or "broken".

As mentioned in the explanation of the dieharder suite in chapter 5.1.2, the software applies the principle of contraposition. The null hypothesis is that the tested generator is an ideal random number generator, which means that it produces an unpredictable and uniformly distributed sequence of numbers. If the test results show a distribution that is significantly different to that of a theoretical, ideal generator, the null hypothesis is considered to be rejected with a certain confidence value—at least $1 - 10^{-6}$ in the case of the dieharder suite. The confidence values are computed by the software and are based on the distribution of individual *pvalues* (see chapter 5.1.2.

If a test shows a distribution that is similar to that of an ideal generator, this does not confirm the null hypothesis. It merely suggests that the generator does not exhibit the weakness that was tested for. Because of this, passed tests are not examined further and the analysis focuses on those tests that fail.

6.2.1 ISO 14443 UIDs

The first test that the sampled UID data fails is the **Diehard Overlapping 5-Permutations Test** with confidence $> 1 - 10^{-8}$. In this test, overlapping

5-tuples of 32-bit integers are taken from the input data and the ordering of their elements is observed—which element is the largest, the second largest, and so on. There are 5! possible orderings in each 5-tuple and the distribution of these orderings over the complete input data is known for an ideal random number generator [30]. Thus it can be compared to the measured distribution of orderings in the input data.

The **Diehard Overlapping 5-Permutations Test** consumes 1 million 32-bit integers per *pvalue* by default, which means that for the 100 *pvalues* generated by the overall test, 400 MiB of input data are read. The collected UIDs however only total a little less than 1.4 MiB, so the result might be biased due to the same numbers being used more than once even within a test for a single *pvalue*. Running the test on sequences of only 100000 integers per *pvalue* leads to a pass although values are still being used more than once during the complete run. Further tests on larger sets of input data should be performed to verify that the generator passes the test using the default parameters, but there is little reason to assume that it will fail.

The **Diehard 6x8 Binary Rank Test** classifies the generator as "weak" using the default parameters, confidence for rejecting the null hypothesis is 0.99688366. The test takes one byte of each of 6 consecutive 32-bit integers, builds a 6x8 binary matrix out of these and determines the rank of that matrix.

A similar explanation as for the Overlapping 5-Permutations Test can be the cause here, as the data passes the test when using half the number of samples per *pvalue* than with the default parameters (50000 instead of 100000). The test consumes 2.4 MiB per *pvalue* and 240 MiB overall. Again, larger sets of input data need to be tested to get a more thorough decision on the quality of the generator.

Exactly the same observation can be made for the **Diehard Count the 1s (stream) Test**, which counts the number of `1` bits in bytes of an overlapping stream provided by the input data. Again, the test classifies the generator as "weak", rejecting the null hypothesis with a confidence of 0.99974415, and halving the number of samples per *pvalue* (128000 instead of 256000) leads to a pass. For a more meaningful test result, the input data should be at least 3.2001 MiB.

The **Diehard Squeeze Test** takes the number $2^{31} - 1 = 2147483647$ and

multiplies it consecutively with floating point numbers from the generator in the interval $[0; 1[$, rounding up intermediate results. It then counts the number of factors n needed to reach 1, as in equation 6.1.

$$\begin{aligned}
&\text{Given a sequence } X \text{ of random numbers} \\
&X = X_1; X_2; X_3; \ldots \in [0; 1[\\
&\text{Define } Y_i \text{ by induction:} \\
&Y_1 = 2^{32} - 1 \\
&Y_{i+1} = \lfloor Y_i * X_i \rfloor \\
&\text{Compute} \\
&n = \min \{j : A_j = 1\}
\end{aligned} \tag{6.1}$$

By default the test finds 100000 such numbers n to generate one *pvalue* and the overall data consumption can not be predicted due to the fact that the number of factors needed depends on the input data itself. The test fails using the default parameters and the *pvalues* are not distributed at all but all of them fall in the interval $]0; 0.1]$. However, searching for only 5000 numbers leads to a passed test although the 1.4 MiB large input file has to be rewound 691 times. The significant change in the distribution of *pvalues* between the two parameter values indicates that a test with a large enough input file (approximately 557 MiB are needed) might lead to interesting results. The confidence for rejecting the null hypothesis here is $> 1 - 10^{-8}$.

Another test that marks the random number generator as "weak" is the **Diehard Runs Test**, rejecting the null hypothesis with a confidence of 0.99888991. It counts the lengths of monotonic decreasing and strictly monotonic increasing sequences of 32-bit floating point numbers taken from the input data and compares their distributions to those of an ideal generator. The distribution of *pvalues* for monotonic decreasing sequences is classified as "weak" whereas that for strictly monotonic increasing sequences passes the test.

The test examines sequences of 100000 numbers for each of the 100 *pvalues* and thus consumes 40 MiB of data. When changing the parameters so that the existing data is only used once (i.e. examining sequences of only 3100 numbers), the test still classifies either distribution as "weak" or even "failed". Using different values for either the sequence lengths or the number of *pvalues* collected often still leads to "weak" or "failed" results. This indicates the possibility of the generator failing the test unambiguously when using a large enough sample size. Further examination of the generator using this test is highly recommended.

The **Diehard Craps Test** plays the dice game craps by using the data to generate dice rolls. 32 bits of data are converted to one dice roll and the number of wins as well as the number of throws necessary to end a game (games are aborted after 21 rolls maximum) are counted. The test thus produces two statistics from a single run, similar to the Diehard Runs Test. In contrast to the Runs test however, both statistics reject the null hypothesis with confidence values of $> 1 - 10^{-8}$ each. Another difference is that smaller numbers of samples—playing 10000 games instead of 200000 to produce one *pvalue*—lead to the test passing the data. As with the other tests, a repetition with a larger sample set should be conducted to verify that the generator in fact does not show a weakness regarding this test, which cannot be safely guaranteed as of now but is expected.

Using the default values, 200000 games are played for each *pvalue* and each can consume up to 42 numbers of 32 bits size (21 rolls with 2 dice). To make sure there is enough data for the test, the sample set should thus include 33.6 MiB of data from the generator.

A result very similar to that of the squeeze test is produced by the **Marsaglia and Tsang GCD Test**, which computes the greatest common divisor of a set of 32-bit integers as well as the number of steps required for the computation using Euclid's Method [10]. While the test passes for small sets of numbers to generate a *pvalue* such as 10000, the *pvalues* are concentrated within the interval $]0; 0.1]$ for the default parameter value of 10 million. Confidence for rejecting the null hypothesis is thus $> 1 - 10^{-8}$ This can be explained by simply computing the data consumation, though. 10 million 32-bit integers equals 40 MiB of data required for the generation of a *pvalue*. This means that in every test run, the same numbers are used (multiple times) and thus the results are obviously the same. Consequently, the test is unlikely to fail when being given large enough sample sets—4 GiB are needed—but this of course is to be verified.

The **STS Monobit Test** is very similar to the Diehard Count the 1s (stream) test with the only difference being that it does not use overlapping bytes. The result is basically the same, the default number of 32-bit values to count the `1` bits in for each of the 100 *pvalues* is 100000 and the test fails with this parameter setting. It rejects the null hypothesis with a confidence of $> 1-10^{-8}$. Using a lower value such as 10000 leads to a passed test and is likely to be the outcome of a test with a large enough data set—4 MiB are sufficient.

Although sharing the same name as the Diehard Runs Test, the **STS Runs Test** has little in common with the former. It counts the lengths of sequences in which all bits are equal. The sequences in which such equal sub-sequences are counted are 400000 bytes in length by default and 100 such sequences are analyzed. The *pvalues* concentrate on the interval $]0; 0.1]$ with a few being within $]0.1; 0.2]$ and the null hypothesis is thus rejected with a confidence of $> 1 - 10^{-8}$. Decreasing the length of the observed sequences to 12400 bytes leads to a "weak" classification and the confidence for rejecting the null hypothesis is 0.99990854. The distribution of *pvalues* also looks visually biased, so another test with the required 40 MiB of sample data is highly recommended and could reveal a weakness in the generator.

The **STS Serial Test** is very intuitive by principle. It determines the distribution of all possible n-bit patterns within a sequence from the generator. It does so using overlapping samples and uses a reference statistic that accounts for the correlation between subsequent samples. The test is executed for values of $n \in [1; 16] \cap \mathbb{N}$, each time 100 sequences of 400000 bytes are sampled by default. Using these values, a lot of sub-tests fail (i.e. for $n \in 1; 2; 3; 4; 5; 12; 15$) or result in a "weak" distribution (i.e. for $n \in 6; 10; 11; 16$), but all subtests pass when using smaller values such as 40000 samples per *pvalue*. It is again unlikely that these tests fail unambiguously when being provided with the required 640 MiB of data, but this cannot be guaranteed.

The author of the dieharder suite implemented a modified version of the STS Serial Test that does not use overlapping samples. This is called the **RGB Bit Distribution Tests**. The parameters are the same except that it is only executed for values of $n \in [1; 12] \cap \mathbb{N}$. Similar to the STS Serial test, a lot of subtests fail with the default values but pass for a sample size of 40000. Correspondingly, the test should be repeated with 480 MiB of input data but will most likely pass as well.

As detailed in chapter 5.1.2, the **Dieharder Generalized Minimum Distance Test** measures the minimum distance of n-dimensional points and compares their distribution to the expected one. Dieharder tests for $n \in 2; 3; 4; 5$ by default, sampling 10000 points for each of the 1000 *pvalues* every time. Using these parameters, the tests fail or give a "weak" result, rejecting the null hypothesis with confidences of 0.99929643 to $> 1 - 10^{-8}$. When using small parameters such as 1000 sample points and 100 *pvalues* per subtest, the collected data suffices and the tests pass. The total of 1.12 GiB of data needed

to run the test with the default parameters differs from the currently available 1̃.4 MiB by several orders of magnitude, but there is little reason to assume that the result would be different.

A test similar to the Diehard Overlapping 5-Permutations Test exists in the dieharder suite, called the **RGB Permutations Test**. The only difference to the test from the original diehard suite is that this one does not use overlapping samples and the tuple size can be varied. By default, it is executed for tuples of size $n \in 2; 3; 4; 5$ and for each subtest, 100 *pvalues* are computed by observing the ordering of 100000 tuples. The test classifies the sample data as "weak" for values of $n \in 3; 4$ and rejects the null hypothesis with confidence 0.99998931 and 0.99999320, respectively. The fact that the test passes the samples for values of $n \in 2; 5$ as well as for all values of $n \in 2; 3; 4; 5$ when observing only 10000 tuples per *pvalue* indicates that this is merely a result of the small input set. A repetition of the test on 40 MiB of sample data is likely to pass.

The **RGB Lagged Sums Test** was designed to search for bit-level correlations in the generated numbers that do not appear until after a certain amount of data has been queried. It adds up 32-bit floating point values in the interval $[0; 1]$ from the data given by the generator, skipping a certain amount n of numbers between subsequent summands. The expected result is $\frac{Numberofsummands}{2}$. The default parameters cause the test to add 1 million samples 100 times, skipping $n = 0$ to $n = 32$ values in between each two. This way, 33 statistics are assembled. The result looks interesting at the first sight, because the distributions of *pvalues* look very similar regardless of the value for n—the *pvalues* are concentrated on the interval $]0.3; 0.5]$. This can however be easily explained when examining the amount of random numbers consumed by the test and the amount of data provided.

1392453 bytes of input data are available and for any parameter n and the test consumes multiples of $4000000 * (n + 1), n \in [1; 33] \cap \mathbb{N}$ bytes for every *pvalue*. This means that the data available for sampling is the same during every *pvalue* computation, the only thing that changes is which subset of this data is taken into account for the sum. There are n possibly different subsets available (only every $(n+1)$th value is added) for each *pvalue*, but 100 *pvalues* are generated. It follows that there are far less possible test outcomes than there are *pvalues* computed, so the reason for the concentration of *pvalues* is obvious. When executing the test with only 100 samples per *pvalue*, the number of possible test outcomes increases to above 100 for all values of n and

the data passes without anomaly. It is likely that the test will succeed with the default parameters when being provided the necessary 211.2 GiB, although the huge difference between the available and the recommended amount of data suggests that this prediction is not necessarily reliable and should be verified with sizes in the order of 500 MiB at least.

Similar to the RGB Bit Distribution Test for 8-bit tuples, the **DAB Byte Distribution Test** measures the distribution of indivdual byte values. Its unique characteristic is that it does so for each byte of a 9-byte-tuple individually. That means that it keeps track of 9 individual byte distributions to search for byte-level correlations in the generator that only appear in regular intervals. The second difference to other tests is that this test produces only a single *pvalue* which also serves as the overall test result. To make sure this value is adequately meaningful, the number of examined byte tuples is very high—51.2 million by default. The length of the tuples can only be varied by recompiling the software suite. While the test rejects the null hypothesis with a confidence of $> 1 - 10^{-8}$, it passes when lowering the amount of sampled tuples to a value of 100000. Because of considerable overhead—the test reads 4 bytes at a time from the generator or file but uses only 3—614.4 MiB of data are needed for a thorough test. The generator is though likely to pass this.

A discrete cosine transformation [1] forms the basis of the **DCT (Frequency Analysis) Test**. It is applied to a block of 256 32-bit values and the largest result is saved. This procedure is repeated on 50000 blocks and the resulting maximum values are tested for uniformity and statistical independence. Just as the DAB Byte Distribution test, it only produces one single *pvalue* as the overall result and the test rejects the null hypothesis with a confidence of $> 1-10^{-8}$ for the default parameters. Running the test on only 5000 blocks of 32-bit values gives no rejection and is the likely outcome of it being run with the required 51.2 MiB of sample data.

The test suite includes 2 tests related to binary trees that function very similar in principle. The first test, called the **DAB Fill Tree Test** fills a fixed-depth binary search tree with 32-bit samples from the generator. The first item is placed at the root node and the subsequent samples are inserted at the child nodes by the well-known rules for binary search trees. As soon as the test would have to place an item in a way that the depth of the tree would rise above the fixed threshold of 32, the insertion is aborted and the current number of nodes in the tree is saved together with the position at which the

insertion failed. By repeating this 15 million times, a large number of samples is gathered. As an additional mechanism to find weaknesses in generators where only certain parts of a 32-bit value are biased, the input values are rotated by cyclic shifting. The positions are then checked for uniformity and the tree sizes are compared with a distribution that was estimated empirically, probably by using a generator that passed all other tests. The latter can of course not be guaranteed to be the exact theoretically ideal distribution, but is considered as such as long as there is no proof of the contrary. This test again fails for the default parameters and rejects the null hypothesis with a confidence of $> 1 - 10^{-8}$, but passes for smaller ones (100000 trees). Due to the dynamic nature, the total consumption of input data using the default values can only be estimated and lies between 1.98 GiB (insertion fails at the 33rd value in each tree) and approximately 257.7 PiB (every tree is filled to the maximum before the insertion fails). It is unrealistic to get enough sample numbers from an Identification Card for the upper bound, but tests using larger samples are highly suggested due to the big difference between the present amount and the lower bound.

The second of the 2 tree-related tests is the **DAB Fill Tree 2 Test**. It starts with an single-node binary tree of fixed depth (128 levels). It then reads bits from the generator subsequently and continues down the left or the right branch, depending on whether the bit is a 1 or a 0, respectively. Whenever going down from a leaf node, a new node is inserted and the path restarts at the root. When an existing node is visited, the read bit is simply discarded. Once the tree would reach a depth of 129, the insertion that would cause this is aborted and again the number of nodes and the position of the failed insertion are saved. In total, 5 million trees are filled by default. Similar to the former test, the tree sizes are compared with an empirically estimated distribution and the positions are tested for uniformity. The results are similar as well, the test fails for the defaults—confidence for rejecting the null hypothesis is $> 1 - 10^{-8}$—and passes for smaller sample sets (25000 trees). The bounds for data consumption are even more extreme here, the lower bound being 80.625 MiB and the upper bound being approximately $2.13 * 10^{44}$ bytes.

The last test of the dieharder suite and the last test that the UID samples fail to pass is the **DAB Monobit 2 Test**. It works, as the name suggests, similar to the STS Monobit test, but it uses small blocks instead of long sequences to count the 1-bits in. The size of the blocks is calculated by the test itself and depends on the generator, the reasons for this are unknown. 65 million blocks

are analyzed by default and the present data lead to a block size of 48 bytes. This nets in a data consumption of 3.12 GiB and as can be expected, the test rejects the null hypothesis. The result is interesting though, because the overall *pvalue* is 1, which means that the result perfectly matches the theoretical distribution of an ideal random number generator. This is indeed no good sign for an actual implementation, because the fact that the generator exhibits a perfectly binomial distribution allows for a predictability of its values. However, the result is entirely different with smaller numbers of examined blocks. For parameters where the provided 1.4 MiB suffice—the test then uses 31000 blocks of 20 bytes each, which is far less than the available amount—the test passes with a *pvalue* of 0.63453322. The implications of these results are not entirely clear, due to the fact that the documentation of the test is scarce and there is not enough data available to run the test with its default parameters.

6.2.2 PACE Nonces

The results of the dieharder suite when being run on the PACE Nonces indicate that the manufacturers possibly use the same or a very similar random number generator as the one responsible for the ISO 14443 UIDs. Many of the tests that the UID data failed—e.g. the STS Monobit Test or DCT (Frequency Analysis) Test—pass the PACE Nonces. Most probably this is due to the much higher amount of data being available. As an example for the reason behind this, the chance of a bit with the value 1 occuring at least once in a sample set of 2 bits is 0.75. The chance of occuring at least once in a sample set of 1000 bits is $1 - 2^{-1000}$. If one now reads 2 bits from a random number generator and these bits both have the value 0, the confidence of rejecting the null hypothesis should be 0.75, which is not enough for the dieharder suite to claim that a weakness has been found. However, running a test that expects a set of 1000 bits on the same data will lead to a rejection of the null hyptothesis with a confidence of $1 - 2^{-1000}$, which is far more than enough for the test to fail although the generator did not change.

Just like earlier, most of the tests that reject the null hypothesis pass if the parameters are adapted to fit the amount of present sample data.

This claim is supported by further examining the result of the **Diehard Squeeze Test**. Again, the *pvalues* are concentrated on the interval $]0; 0.1]$

when using the default parameters and again the test passes when searching for only 5000 values. The recommendation is likewise, a large enough sample set should be tested to gain certainty of the question whether this is due to a weakness in the design or implementation.

The **Diehard Runs Test** also rejects the null hypothesis for the PACE Nonces using the default parameters, the confidence is 0.99972851. However, in contrast to the anomaly observed with the UID samples, the test passes when using sequences of at least 9000 elements and variation of the length does not lead to "weak" results when keeping it above 9000. The noteworthy behaviour observed during the tests on the ISO 14443 UIDs does occur when using sequence lengths of less than 9000. This indicates that the effects are actually an effect of the small sample size, provided that the two random number generators really are the same.

A major difference in test results between the two sources is reported by the **STS Runs Test**. While it rejected the null hypothesis rather consistently for the UID samples, the PACE Nonces consistently pass even for sequence lengths as small as 1000. This can mean one of two things. Either the two sources are supplied by two different random number generators or the remarkable results of the UID analysis is due to the sample data appearing biased by chance, which is entirely possible. As of now, it seems that the second statement is more probable, since the other test results rather suggest that the two generators are the same.

The remaining test results are, as stated, similar to those of the UIDs. There is not enough sample data available to run most of the tests with their default parameters, but running them with accordingly smaller values suggests that the generator does not show any of the weaknesses the suite checks for.

6.3 Timing Analysis

Using the newly implemented capabilities of the Proxmark III Toolkit, timings of the PACE protocol were sampled using replayed messages of previously captured executions.

An execution of the PACE protocol consists of 5 APDU-pairs being exchanged

between the reader device and the card—each consisting of a Command APDU and a Response APDU. When using replayed messages, the card responds to the 4th command APDU with an error message and prevents the replay of the 5th command APDU. The first two APDU pairs mereley set up the protocol and transfer an encrypted nonce, no information specific to the card at hand is included in them. Thus, it only makes sense to measure the timings of the 3rd and 4th APDU pairs.

For this analysis, a total of 12 Electronic Identification Cards have been used, denoted here as cards A to L. Card F is the sample already used as data source for the random number analysis and the others are Identification Cards owned by existing persons. A successful execution of the PACE protocol has been performed with cards A to F using GlobalTester [20] and the APDU data has been extracted from the log files. The Proxmark III Toolkit has then been used to replay the collected APDUs to cards A and B and the relevant timings have been recorded. Card A was subjected to replayed APDUs of cards A, C, D, and E and card B was subjected to replayed APDUs of cards B, C, D, and E. The cards that the APDUs were taken of are called *source cards* in the following paragraphs and the cards the these APDUs were sent to are called *target cards*. For every pair of target and source card 100 timing samples were taken to account for deviation caused by the communication medium.

The timing data has been visualized using box plots. Red horizontal lines show the median values and blue boxes indicate the interval in which 50% of the samples are located. Black bars outside these boxes reach to the absolute minimum and maximum values.

6.3.1 Map Nonce APDU timings Per Card Pair

At first, the `Map Nonce` APDU timings were gathered. Figure 6.3 shows their distributions.

As one can see in the plot, there are differences in the timings between APDUs from the same card (A-A and B-B) and from different cards (A-C, B-C etc.). However, the differences can not be used to identify a card, because they take negative as well as positive values and seem to have no apparent correlations. When being presented this plot without the labels on the X-axis, one could not identify which timings belong to a matching card pair (A-A or B-B) and

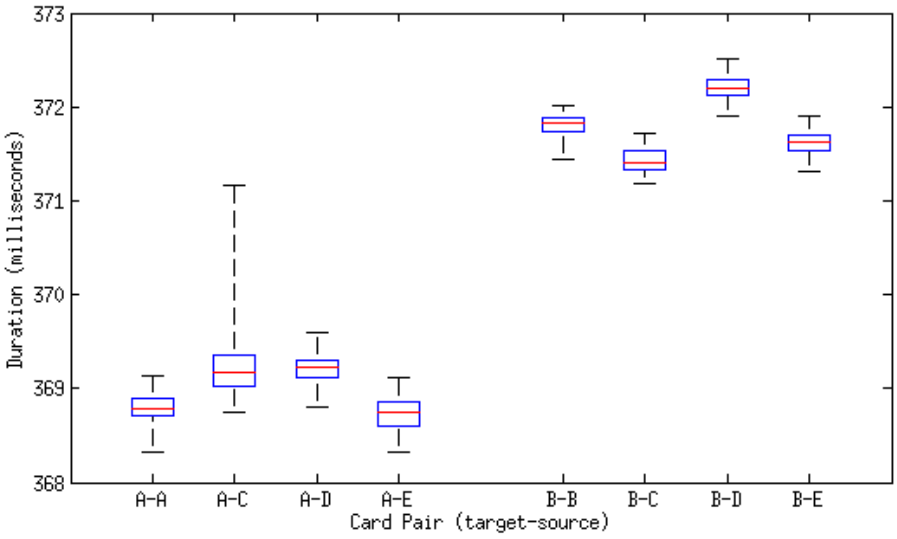


Figure 6.3: *The timings of replayed* `Map Nonce` *APDUs. The times were measured from the point where the Proxmark starts sending the command APDU to the point where it finished receiving the response APDU.*

which do not.

Although the difference of timings for one target card (e.g. A-A, A-C, A-D, A-E) do not seem to allow the identification of the card, it is obvious that the two different target cards A and B have a rather large difference in their overall timings. A detailed examination of this follows.

6.3.2 Perform Key Agreement APDU Timings Per Card Pair

The second pair of APDUs, namely for the `Perform Key Agreement` step of the protocol was given the same timing analysis. Figure 6.4 shows their distributions.

The plot shows similar characteristics to that of the `Map Nonce` APDUs. While there are differences in the timings for each target card, these are not recognizable without the knowledge of the labels and thus do not allow an identification

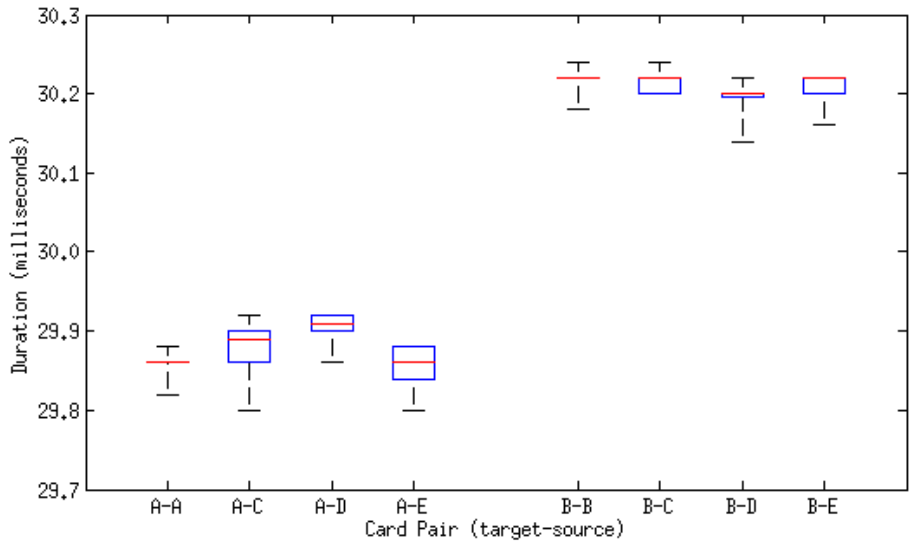


Figure 6.4: *The timings of replayed* `Perform Key Agreement` *APDUs. The durations were measured in the same way as the* `Map Nonce` *APDUs.*

of the card based on these features. However, similarly to the first timing analysis, a relatively large difference lies between the overall timings of the two target cards. This suggests that cards might be identifiable by their response time to any given set of replayed command APDUs, regardless of their source.

6.3.3 Overall Response Times Per Target Card

To find out whether the observed timing patterns are unique for one card among a larger set, Every available card was subjected a timing analysis using both APDUs of a communication with the card itself as well as APDUs from different cards. For each card, the timings of all 4 performed PACE steps were recorded, since the timing characteristics do not seem to be related to the content of the APDUs, as discovered in chapters 6.3.1 and 6.3.2. Figure 6.5 shows the measured times.

There is a special property of the timings that is not clearly visible in the figure but can be seen in the numerical data which has been exemplarily summarized

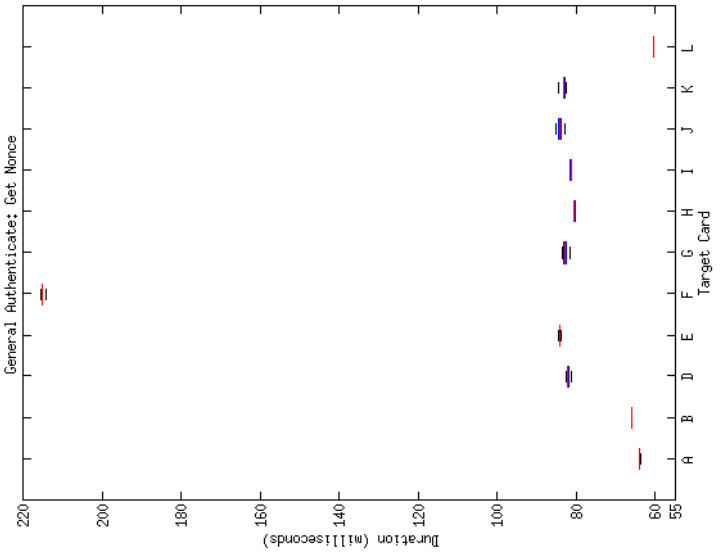
in table 6.1 for the `Get Nonce` step. The intervals between maximum and minimum durations overlap for some cards (e.g. cards E and J for the `Get Nonce` step), but this similarity is not consistent through all steps. As figure 6.5 shows, cards E and J show quite different characteristics during the `MSE: Set AT` step.

Target card	Median	Arithmetic mean	Minimum	Maximum
A	63898 μs	63882 μs	63778 μs	63918 μs
B	65856 μs	65847 μs	65756 μs	65876 μs
D	82086 μs	82047 μs	81256 μs	82596 μs
E	84076 μs	84072 μs	83926 μs	84556 μs
F	215328 μs	215261 μs	214298 μs	215498 μs
G	101434 μs	101094 μs	96126 μs	103584 μs
H	80406 μs	80382 μs	80086 μs	80596 μs
I	81296 μs	81283 μs	81036 μs	81526 μs
J	84056 μs	84085 μs	82936 μs	85186 μs
K	83076 μs	83081 μs	82656 μs	84376 μs
L	60268 μs	60276 μs	60228 μs	60308 μs

Table 6.1: *Numerical timing data for the `Get Nonce` step. As can be easily seen, the deviation intervals of the different cards rarely overlap, which results in a high entropy of these timing characteristics.*

To mathematically express the timing characteristics of a single card, one can write the deviation intervals as two vectors. The vector $min = (min_1; min_2; min_3; min_4)$ consists of the minimum timings and the vector $max = (max_1; max_2; max_3; max_4)$ of the maximum timings of the four individual steps. As an example for card L, according to table 6.1, $min_2 = 60228$ and $max_2 = 60308$. To check whether a newly measured set of four timings v of an unknown card matches a specific card of which the characteristics are known, one would then compute the differences $max - v$ and $max - min$. If each element of $max - v$ is less than or equal the corresponding element of $max - min$, all measured timings lie within the ranges that were observed for the known card. This means that the newly measured card is probably the same as the known.

Given the fact that card F is a sample card that is not in productive use, existing cards seem to show response times within a range of approximately 30 to 120ms, depending on the step. The deviations from the median for one card and one step seem to be no more than 1ms usually.


General Authenticate: Get Nonce
Duration (milliseconds)
220
200
180
160
140
120
100
80
60
55
A B D E F G H I J K L
Target Card

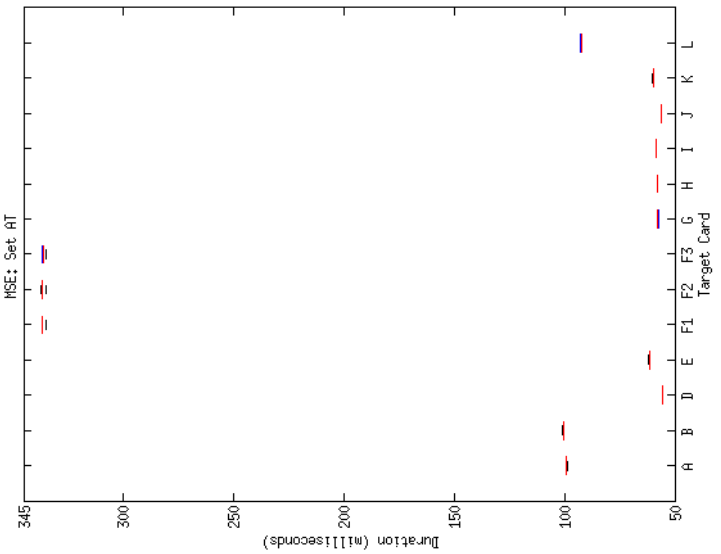

MSE: Set AT
Duration (milliseconds)
345
300
250
200
150
100
50
A B D E F1 F2 F3 G H I J K L
Target Card

Figure 6.5: *The measured timings for different Identification Cards. The boxes and absolute minimum/maximum values are barely visible for some cards due to the large scale. This however shows that the cards are easily recognizable by their timing patterns and that the deviations are relatively small compared to the differences between the individual cards. For the* `MSE: Set AT` *step, card F has been tested three times by placing it next to the Proxmark III's antenna in different ways. This was done to see whether the placement of antenna and card has any influence on the timings. The figure as well as the numerical data show that this is not the case.*

Theoretically, 15 to 60 non-overlapping deviation intervals per step are thus possible. Taking into account for all four steps, approximately 2 million distinct timing characteristics (in the form mentioned above) are possible. The 2 million distinct characteristics would allow for a certain recognition of a card by examining only a single execution of a PACE replay. If an attacker has the time to perform multiple of these replays, he can even estimate the deviation intervals and overlapping ranges between different cards are less of a problem. This would further increase the chances of success for an attack. During the test, no two cards showed the same characteristics in all four steps. This indicates that a tracking attack is in fact possible.

While the numbers look extremely high at the first sight, one has to consider the generalized birthday problem [13] when calculating the chances of positively recognizing a previously seen cards. If there are 2 million distinct timing characteristics, the chance of two cards showing the same in a set of 2000 are already above 60%. Within a set of 1000, it is still above 20%.

This observation however shows that the recognition of a previously seen Identification Card is possible within a set of several hundred with only a small chance of a false positive. The exact numbers of the actually existing distinct timing characteristics can be estimated with a much larger set of Identification Cards to sample from, but the experiments so far indicate that they indeed lie within the range of several hundred thousands.

Chapter 7

Summary

This chapter summarizes the results of the analysis, explains the implications of these findings and gives perspectives as well as recommendations for future work on this research field.

7.1 Summary of Analysis Results

The random number generators employed in the new German Identification Cards seem to offer randomness and unpredictability well enough to thwart attempts to trace cards based on previously observed patterns or to even mount attacks on the cryptographic protocols. There are a few inconclusive test results and most of the tests should be repeated with much larger sample sets, but overall the strength of the generators can be considered adequate. They have passed many tests that are known to have shown weaknesses in other generators and that test for a large variety of different properties of the produced values. It is assumed that both the ISO 14443 UIDs as well as the PACE nonces are produced by the same random number generator, due to the largely similar results and the benefits of this solution in form of less work, less risk of errors in the implementation and thus a lower production price for manufacturers.

The timings of replayed APDU batches did not reveal a weakness like the one

discovered in the implementation of the BAC protocol [11], but a different flaw has been uncovered within the course of the analysis. Different cards seem to exhibit different timing characteristics alltogether, regardless of the content of the transmitted APDUs. This allows an attacker to sample timing data from a victim's Identification Card and later identify the card remotely via the observed response durations among a set of several hundred other cards. The only thing to do is to replay APDUs of a PACE handshake to any cards in the range of a reader device and observe the response times. Once a card shows similar response times to that of the victim, it is likely to be the same. No knowledge about the victim at all is required for this method and the hardware needed to perform it is available as well as affordable for almost any person. The chance of error when using this method can be computed in advance and introduces the danger of possible adversaries who scale their attacks based on the needed precision.

7.2 Perspectives and Recommendations for Future Work

There are multiple possibilities of continued work on traceability of electronic documents. One of them is the confirmation and explanation of the findings of this thesis. For example, the anomalies found in the timing patterns of Identification Cards can be verified and completely determined using larger sets of cards and laboratory-grade equipment. Features like ambient temperature can influence the computational power of the RFID chip and influence the response times. Access to implementation details of the RFID chip's software or a cooperation with the BSI and manufacturers can help make certain that all aspects are accounted for.

If the claims made here are found to be valid, the precise consequences of them can be identified. The size of sets that allow for a recognition of previously seen cards as well as the exact error rates of this attack vector can be worked out.

Once the numerical properties of the attack are known, it can be combined with other traits of the Identification Card or with similar attacks on other RFID chips to further decrease the chances of error. The card type distinction

mentioned in chapter 3.2 does not offer enough entropy to allow a traceability attack alone, but knowledge gained by it can be used in other attacks to boost their precision and reliability. The entropy is assumed to be in the range of 2–5 bits, due to the fact that only three different types of cards observed during the tests and there is no gain for the manufacturers to introduce many differences. Different RFID chips that are carried around by people can also offer similarly useful information.

Lastly, the same attack can be applied to other chips, such as a bank card's contactless payment option which is becoming more and more popular in Germany at the moment [2]. RFID chips using cryptographic protocols are especially vulnerable to this, as different cards can use different encryption keys that in turn can have influence on the computational effort needed to respond to commands and thus affect the response times of the device.

Bibliography

[1] N. Ahmed, T. Natarajan, and K.R. Rao, *Discrete Cosine Transform*, IEEE Transactions on Computers **C-23** (1974), no. 1, 90–93.

[2] Eva Allar, *GeldKarte kontaktlos: Pünktlich zur 2. Halbzeit dank Chip*, March 2009, available at `https://www.geldkarte.de/_www/de/pub/geldkarte/presse/presse-informationen/pressemitteilungen/archiv/i9589_1_pm_geldkarte_kontaktlos_bayer04.php`, accessed 08.11.2012.

[3] *`https://www.ausweisapp.bund.de`*, online, accessed 13.10.2012.

[4] Harald Baier and Tobias Straub, *Vom elektronischen Reisepass zum Personalausweis: RFID und personenbezogene Daten – Lessons Learned !?*, Im Focus das Leben, Beiträge der 39. Jahrestagung der Gesellschaft für Informatik e.V., Lecture Notes in Informatics, vol. P-154, Gesellschaft fur Informatik e.V. (GI), September 2009, pp. 1717–1731.

[5] Mike Bond, Omar Choudary, Steven J. Murdoch, Sergei Skorobogatov, and Ross Anderson, *Chip and Skim: cloning EMV cards with the pre-play attack*, ArXiv e-prints (2012).

[6] Robert G. Brown, *Dieharder: A Gnu Public License Random Number Tester*, included as `manual/dieharder.tex` in the dieharder sources available at `http://www.phy.duke.edu/~rgb/General/dieharder/dieharder-3.31.1.tgz`, accessed 23.10.2012.

[7] Robert G. Brown, Dirk Eddelbuettel, and David Bauer, *Dieharder: A Random Number Test Suite*, online, available at `http://www.phy.duke.edu/~rgb/General/dieharder.php`, accessed 13.10.2012.

[8] Bundesamt für Sicherheit in der Informationstechnik, *Technische Richtlinie TR-03116-2, eCard-Projekte der Bundesregierung, Teil 2 — Hoheitliche Ausweisdokumente*, 2012.

[9] Dario Carluccio, Kerstin Lemke-Rust, Christof Paar, and Ahmad-Reza Sadeghi, *E-Passport: The Global Traceability Or How to Feel Like a UPS Package*, Information Security Applications (Jae Lee, Okyeon Yi, and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 4298, Springer Berlin / Heidelberg, 2007, pp. 391–404.

[10] Lindsay N. Childs, *Euclid's Algorithm*, A Concrete Introduction to Higher Algebra (Sheldon J. Axler and Ken Ribet, eds.), Undergraduate Texts in Mathematics, Springer New York, 2009, pp. 27–52.

[11] Tom Chothia and Vitaliy Smirnov, *A Traceability Attack Against e-Passports*, Financial Cryptography and Data Security (Radu Sion, ed.), Lecture Notes in Computer Science, vol. 6052, Springer Berlin / Heidelberg, 2010, pp. 20–34.

[12] Boris Danev, Thomas S. Heydt-Benjamin, and Srdjan Čapkun, *Physical-layer Identification of RFID Devices*, USENIX Security '09: Proceedings of the 18th USENIX Security Symposium.

[13] Anirban DasGupta, *The Birthday and Matching Problems*, Fundamentals of Probability: A First Course, Springer Texts in Statistics, Springer New York, 2010, pp. 23–28.

[14] Bundesministerium des Innern, *Neuer Personalausweis*, online, available at `http://www.bmi.bund.de/DE/Themen/Sicherheit/PaesseAusweise/ePersonalausweis/ePersonalausweis_node.html`, accessed 13.10.2012.

[15] *`http://stat.fsu.edu/pub/diehard/`*, accessed 13.10.2012.

[16] Whitfield Diffie and Martin E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory **22** (1976), no. 6, 644–654.

[17] Yasmin El-Sharif, *Niederländische NXP: Oranje sichert sich deutschen Perso-Chip*, Spiegel Online, August 2010, available at `http://www.spiegel.de/wirtschaft/unternehmen/niederlaendische-nxp-oranje-sichert-sich-deutschen-perso-chip-a-712655.html`, accessed 13.10.2012.

[18] Mark Fischler, *Distribution of minimum distance among N random points in d dimensions*, Tech. Report FERMILAB-TM-2170, Fermi National Accelerator Lab., Batavia, Illinois, May 2002.

[19] Bundesamt für Sicherheit in der Informationstechnik, *Technical Guideline TR-03110, Advanced Security Mechanisms for Machine Readable Travel Documents*, October 2010.

[20] *http://globaltester.org*, accessed 21.10.2012.

[21] *ICAO Doc 9303, Machine Readable Travel Documents — Part 1, Machine Readable Passports — Volume 2, Specifications for Electronically Enabled Passports with Biometric Identification Capability*, 2006.

[22] *ICAO Doc 9303, Machine Readable Travel Documents — Part 3, Machine Readable Official Travel Documents — Volume 2, Specifications for Electronically Enabled MRtds with Biometric Identification Capability*, 2008.

[23] *ISO/IEC FCD 14443-1 — Identification cards — Contactless integrated circuit(s) cards — Proximity cards — Part 1: Physical characteristics*, February 2007.

[24] *ISO/IEC FDIS 14443-2 — Identification cards — Contactless integrated circuit(s) cards — Proximity cards — Part 2: Radio frequency power and signal interface*, July 2009.

[25] *ISO/IEC FCD 14443-3 — Identification cards — Contactless integrated circuit cards — Proximity cards — Part 3: Initialization and anticollision*, November 2008.

[26] *ISO/IEC FCD 14443-4 — Identification cards — Contactless integrated circuit(s) cards — Proximity cards — Part 4: Transmission protocol*, March 2007.

[27] *ISO/IEC 7816-4 — Identification cards — Integrated circuit cards — Part*

4: Organization, security and commands for interchange, 2005.

[28] Yifei Liu, Timo Kasper, Kerstin Lemke-Rust, and Christof Paar, *E-Passport: Cracking Basic Access Control Keys*, On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS (Robert Meersman and Zahir Tari, eds.), Lecture Notes in Computer Science, vol. 4804, Springer Berlin / Heidelberg, 2007, pp. 1531–1547.

[29] George Marsaglia, *Diehard Test Descriptions*, available at `http://www.stat.fsu.edu/pub/diehard/cdrom/source/tests.txt`, accessed 20.10.2012.

[30] ———, *A Current View of Random Number Generators*, Computing Science and Statistics: Proceedings of the 16th Symposium on the Interface (Amsterdam, Netherlands), Elsevier Science Publishers B.V., 1985, pp. 3–10.

[31] George Marsaglia and Wai Wan Tsang, *Some Difficult-to-pass Tests of Randomness*, Journal of Statistical Software **7** (2002), no. 3, 1–9.

[32] George Osipenko, *Attractors*, Dynamical Systems, Graphs, and Algorithms, Lecture Notes in Mathematics, vol. 1889, Springer Berlin / Heidelberg, 2007, pp. 65–83.

[33] Senthilkumar Chinnappa Gounder Periaswamy, Dale R. Thompson, Henry P. Romero, and Jia Di, *Fingerprinting Radio Frequency Identification Tags Using Timing Characteristics*, Radio Frequency Identification System Security: RFIDsec'10 Asia Workshop Proceedings, Cryptology and Information Security, vol. 4, February 2010, pp. 73–82.

[34] *http://www.proxmark.org*, online, accessed 13.10.2012.

[35] *http://code.google.com/p/proxmark3/wiki/HomePage*, online, accessed 13.10.2012.

[36] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, Special Publication 800-22, available at `http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf`, accessed 13.10.2012,

April 2010.

[37] Jens Witte, *Regierungsauftrag: Infineon stellt Chips für neuen Perso her*, Spiegel Online, December 2010, available at `http://www.spiegel.de/wirtschaft/unternehmen/regierungsauftrag-infineon-stellt-chips-fuer-neuen-perso-her-a-733293.html`, accessed 13.10.2012.

[38] Michal Zalewski, *Strange Attractors and TCP/IP Sequence Number Analysis*, online, 2001, available at `http://lcamtuf.coredump.cx/oldtcp/tcpseq.html`, accessed 13.10.2012.

Ehrenwörtliche Erklärung

Ich versichere hiermit, dass die vorliegende Masterarbeit mit dem Titel "An Analysis of Traceability of Electronic Identification Documents" von mir selbstständig, ohne Hilfe Dritter und ausschließlich unter Verwendung der angegebenen Quellen angefertigt wurde. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, habe ich als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form, auch in Teilen, keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Declaration of Authorship

I hereby declare that I have developed and written the enclosed master's thesis entitled "An Analysis of Traceability of Electronic Identification Documents" entirely on my own and have not used outside sources without declaration in the text. Any concepts or quotations applicable to these sources are clearly attributed to them.

This master's thesis has not been submitted in the same or substantially similar version, not even in part, to any other authority for grading and has not been published elsewhere.

Ort, Datum Place, Date	Unterschrift Signature