University of Tartu
Institute of Computer Science
Computer Science Curriculum

Hain Luud

An analysis of the HID® Indala and Seos™ protocols

Bachelor's Thesis (9 ECTS)

Supervisor: Danielle Morgan

Tartu 2021

An analysis of the HID® Indala and Seos™ protocols

Abstract:

This thesis aims to give a more detailed and comprehensive overview of the two most used door access control technologies in the University of Tartu buildings: Indala and Seos™, than what is currently publicly available.

The wireless communication protocols, memory layout, and how they hold up to common RFID card vulnerabilities were studied to analyse these technologies. Additionally, a Java card applet was developed to assist further research into Seos™ and other high-frequency cards.

The research identified that both technologies use very different methods for card authentication, have contrasting memory layouts and operate on two distinct frequencies. One significant finding was that Indala cards are susceptible to cloning and replay attacks. Additionally, it was found that the average Levenshtein distance of the 13 Indala 224-bit UIDs collected for research was 12 nibbles.

Another important finding was that the RNG used in a Seos™ card might be weak. In an experiment comparing a large number of UIDs generated both by using the card and the Python Numpy library, the Seos™ dataset exhibited a statistically unlikely amount of UID collisions.

Keywords:

RFID, access card, Indala, Seos, HID, Proxmark, Java card, smart card, RNG

CERCS: P175

HID® Indala ja Seos™ protokollide analüüs

Lühikokkuvõte:

Selle lõputöö eesmärk on anda detailsem ja ulatuslikum ülevaade kahest Tartu Ülikooli hoonetes enim kasutusel olevast kiipkaarditehnoloogiast, Indala ja Seos™, kui on hetkel avalikult kättesaadav.

Tehnoloogiate analüüsimiseks uuriti kiipkaartide kommunikatsiooniprotokolle, mälustruktuuri ja nende vastupidavust tüüpilistele haavatavustele. Seos™ ja teiste kõrgsagedusel töötavate kaartide uurimise toetamiseks loodi ka Java aplet.

Töö käigus leiti, et mõlemad tehnoloogiad kasutavad erinevaid autentimismeetodeid, mälustruktuure ning sagedusi. Töö üheks väljapaistvaks avastuseks oli, et Indala kaardid on haavatavad kloonimis- ja taasesitusrünnetele. Lisaks leiti, et uurimise käigus kogutud 13 Indala 224-bitise UID keskmine Levenshteini kaugus oli 12 poolbaiti.

Lisaks avastati töö käigus, et Seos™ kaartides kasutusel olev juhuarvude generaator võib olla nõrk. Katses võrreldi kaht suurt valimit - üks genereeritud Seos™ kaardiga ja teine arvutis Pythoni Numpy teegiga. Võrdluse käigus ilmnes, et Seos™ kaardi loodud valimis oli statistiliselt ebatõenäoline arv põrkeid, kui võiks eeldada juhuslikult genereeritud arvude puhul.

Võtmesõnad:

RFID, kiipkaart, Indala, Seos, HID, Proxmark, Java aplet

CERCS: P175

Table of Contents

Introduction

One of the modern ways to regulate access to rooms or buildings is wireless or contactless keycards. These come in many different shapes and sizes, but one of the more common form factors is the ID-1 type card which has its dimensions and additional characteristics defined in the ISO/IEC 14443-1 and 7810 standards [1, 2]. An ID-1 type card is 85.5mm * 54mm, usually made of plastic and stores a digital or physical pattern that the door access system recognises.

There are several types of keycard implementations: magnetic stripe cards, Wiegand wire embedded cards, smart cards, and RFID proximity cards. These proximity keycard implementations are prevalent and used in many buildings throughout the world. However, due to its widespread adoption as the new and convenient access system and the operating procedure being invisible to the naked eye, blind spots that customers do not see and vendors ignore could lead to a particular technology being exploited.

This thesis will focus on the RFID proximity card technologies most used in the buildings of the University of Tartu (UT). Since there do not seem to exist any publicly disclosed detailed descriptions of either Indala or Seos™, this research aims to give a more detailed and comprehensive overview of them than the information currently publicly available.

The thesis is divided into three parts. Chapter 1 will give an overview of the RFID technology in general and describes the memory and communication protocols of the two specific technologies implemented in the UT buildings. Chapter 2 will present what tests were conducted to research the card technologies. The results and what information could be gained from these experiments are laid out in Chapter 3.

Terms and Definitions

ADF (Application Dedicated File) - An individual container within the Seos™ Vault that is used to store one or more digital credentials [3].

Anticollision loop – An algorithm used to set up the dialogue order between a PCD and one or more PICCs [4].

BER-TLV (basic encoding rules - tag, length, value) – encoding scheme for data objects. The encoding scheme represents data objects using three fields: tag, length and value [5].

Brute-forcing attack - An attack where all possible combinations are tried to find the right match [6].

Cloning attack – A card cloning attack is a process in which data from a legitimate RFID card is captured and then a new unauthorised copy is created [6].

Levenstein distance – A measure frequently used to calculate the difference of two strings based on the number of minimal edits one needs to do to convert one string to another. [7]

Nonce – a random or pseudo-random number that is used in cryptographic communication protocols to prevent replay attacks. The nonce value is usually generated and attached to messages or request so that the recipient can verify the message's authenticity [8]

Replay attack – In a replay attack, an attacker captures a valid RFID signal from intercepting communication between a reader and a tag to replay it back to the system at a later time [6].

RFID (radio frequency identification) – A technology that uses electronic tags placed on objects, people, or animals to relay identifying information to an electronic reader using radio waves [9].

Secure messaging/channel – cryptographically secured data transmission between a card and the outside world [10].

Seed - A starting point number for a random number generator [11]

Seos™ Authentication Key Set - The Key Set required to read the digital credential from an ADF. The Key Set comprises an encryption key and MAC key, which collectively establish the Secure Channel [3].

Seos™ Edge – The interface exposed by the Seos™ vault that enables an authorised system to read and write the card's memory [3].

Seos™ Vault – A software application executing on a physical credential that takes responsibility for the secure storage and use of digital credentials [3].

SIO (Secure Identity Object) - An instance of a digital credential that is compliant with a specific HID defined format. The format provides mechanisms to encrypt the ID, sign the digital credential and bind it to a physical credential [3].

Smart card – a card with an integrated embedded circuit in its body. The card has components for transmission, processing and storing data and contacts on its surface or an electromagnetic field for data transmission. [10]

UID – A unique ID used to identify a card or a number needed for the anticollision algorithm defined in the ISO/IEC 14443-3 standard [12].

Abbreviated terms

APDU – application protocol data unit

ATQ – answer to request

ATR – answer to reset

ATS – answer to select

DF – dedicated file

MF – master file

PCD – proximity coupling device (reader)

PICC – proximity card

PPS – protocol and parameter selection

RATS – request for answer to select

RNG – random number generator

RRG – RFID Research Group

SAK – select acknowledge

WUPA — wake-up command for PICC type A

1. Background

In RFID based access control systems, a keycard works as a key or security token, enabling access through electrically powered doors when tapped on the reader mounted next to them [13].

RFID cards consist of a microchip, a coiled antenna used to transmit data to the reader and the plastic card itself [14]. A battery inside the card also determines if it is a passive or an active card. Most keycards are passive, including the ones in focus of this paper, meaning that they get energy from the reader to power their circuit [10].

Umar Farooq *et al.* [15] explain that the keycard transmits information stored in its microchip to the reader when it comes in the vicinity of the electromagnetic field generated by the reader. The phenomenon is based on Faraday's law of electromagnetic induction. The current flowing through the reader's coil produces a magnetic field that links to the keycard's coil antenna, thereby producing a current in the card. The keycard then varies this current by changing the load on its antenna. This variation is then picked up by the reader and decoded into bits. The data is then sent through a communication interface to a host computer system that processes the requests. Once the host system recognises the access credentials, it unlocks the door [15].

RFID cards also come in different frequency categories: Low Frequency (LF), High Frequency (HF) and Ultra High Frequency (UHF). Most often, 125KHz, 13.56 MHz and 433 MHz frequencies are respectively used. Of the three, UHF is the least used category, and in general, LF cards do not have security standards and are more expensive than HF cards [16].

An Electronics Notes article [17] explains that previously there were many generations of RFID access systems that did not use any security implementations against attacks. Many such cards are still in use today, but some have taken steps to integrate security into the authentication process. According to the article, the two main approaches for preventing cloning attacks are rolling code and challenge-response authentication [17].

In the rolling code implementation, the card changes its UID after each interaction with the reader. Previous codes are also ignored to prevent replay attacks. New codes will be generated using pseudorandom number generators [17].

The article [17] further clarifies that with the challenge-response authentication approach, the reader issues an enquiry to the tag, which results in a response. However, the digital

credential is never sent over the interface between the reader and the card. Instead, only encrypted data is sent over radio analogously to public-key encryption [17]. The rest of this chapter focuses on the two technologies most used for physical access control in the UT buildings and a third technology that was used for testing purposes covered in Chapter 2.

1.1 Indala

The Indala Proximity series cards transmit on a 125KHz low-frequency range and are a popular HID® Global product. The technology is proprietary and no detailed information has been shared about its memory or communication protocol. Therefore, this section only details information originating from a few whitepapers [18, 19, 20] put out by HID® Global. More information about this technology is reported in Chapter 3 based on the findings of the tests described in Chapter 2.

1.1.1 Memory

For most of the Indala line cards, the datasheet [20] lists their memory as being N/A. However, other combined technology variations such as Indala FlexISO MIFARE cards use from 1kB to 16kB depending on the more advanced technology's type and requirements [20]. The combined technology variants will not be analysed because they are not in use in the UT facilities. The memory of an Indala card is not writable, and there currently is no good way to inspect its contents.

1.1.2 Communication protocol

An Indala technology whitepaper [19] explains that a card starts transmitting its data in a predefined format when in range of a reader. There are multiple data formats available for HID® Global customers but by default Indala cards and readers use the 26-bit Wiegand data encoding format also depicted in Figure 1.

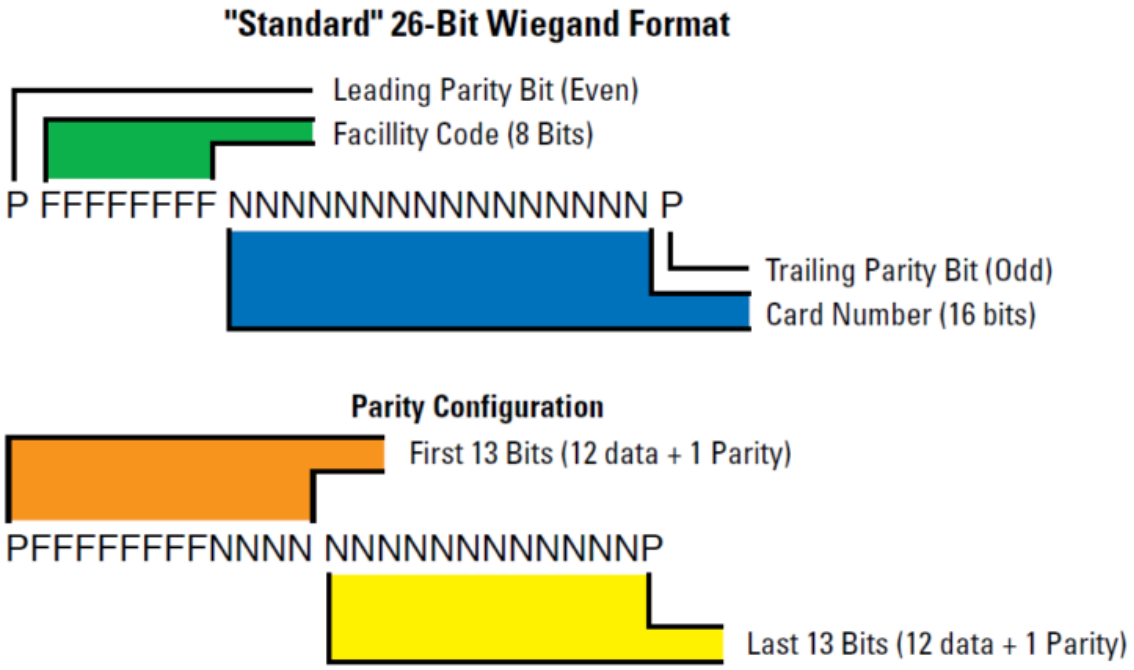

Figure 1. Wiegand format [21].

The reader intercepts the data, demodulates it and sends the data over to the access controller.

Some Indala cards and readers use an additional security measure named FlexSecur®. According to the whitepaper [19], this technology creates a data format for individual customers and encrypts the data on the Indala card. The card starts transmitting its encrypted data according to the new format upon entering an electromagnetic field. After receiving the encrypted data, a FlexSecur® reader locates the password within the card's data and checks whether the password on the card matches its own. Figure 2 illustrates this verification process. All Indala readers that operate in the same facility store the same password. If a match is made, the reader decrypts the data and sends the access control data to the access control panel.

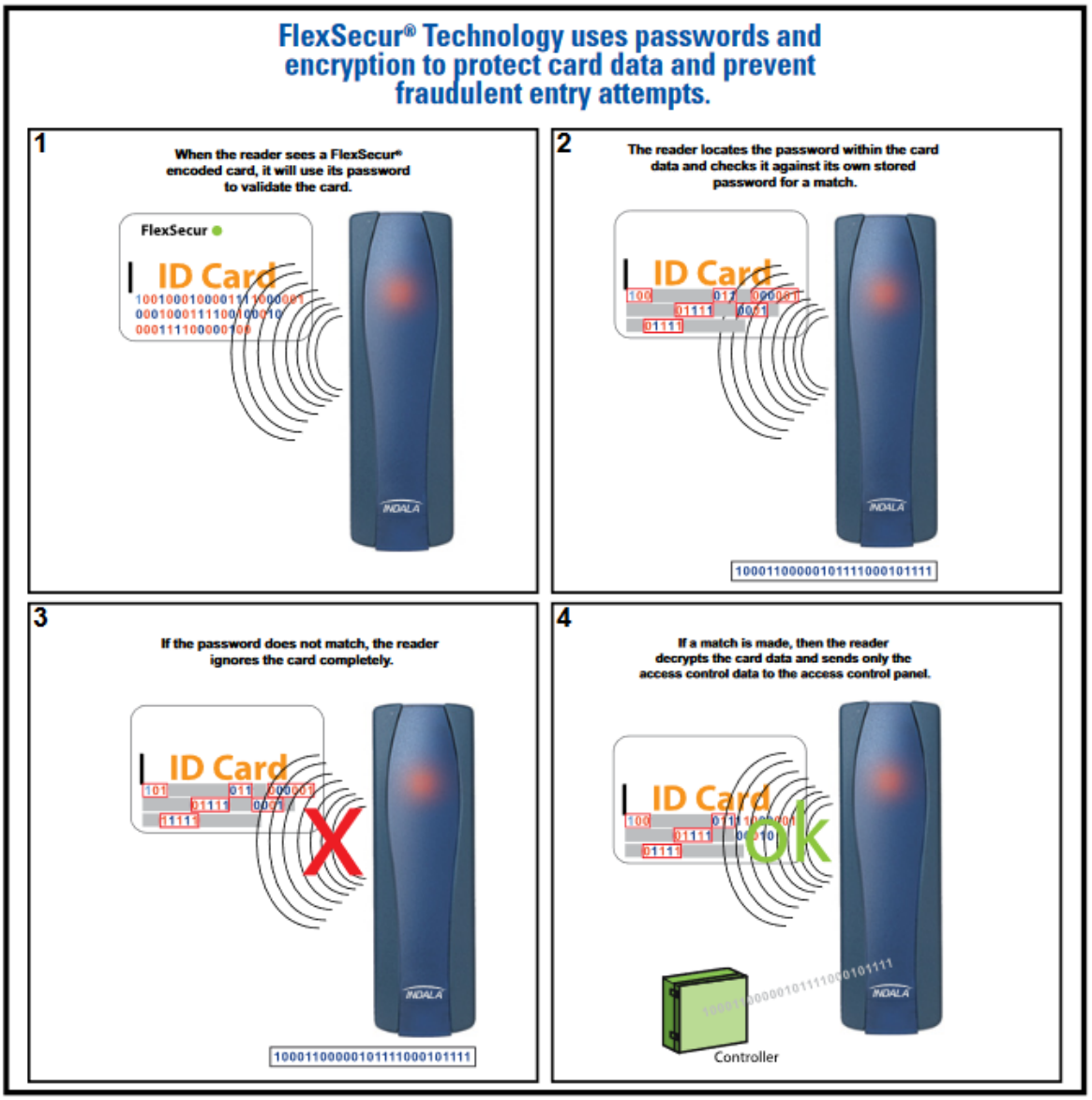

Figure 2. An illustration of the Indala verification process [19].

Customers can order cards that use a proprietary Original Equipment Manufacturers (OEM) format or a unique data format created only for them [19]. The added security comes in the form of security by obscurity because OEM formats may be less known and could therefore be harder to decode. A longer format may provide more resistance to brute-forcing attacks as well.

1.2 T5577

A T5577 card uses an Atmel ATA5577 microchip and operates on the 125KHz frequency [22]. The technology is easily programmable and customisable. Though no T5577 cards are

used in the UT buildings, knowing how this technology works allows one to derive information about the Indala technology. How these two technologies relate to each other and what can be concluded from the relationship is covered in Chapter 2 of the thesis.

1.2.1 Memory

An Atmel ATA5577 chip has 363-bit EEPROM memory, which is structured in 11 blocks of 33 bits, as shown in Figure 3.

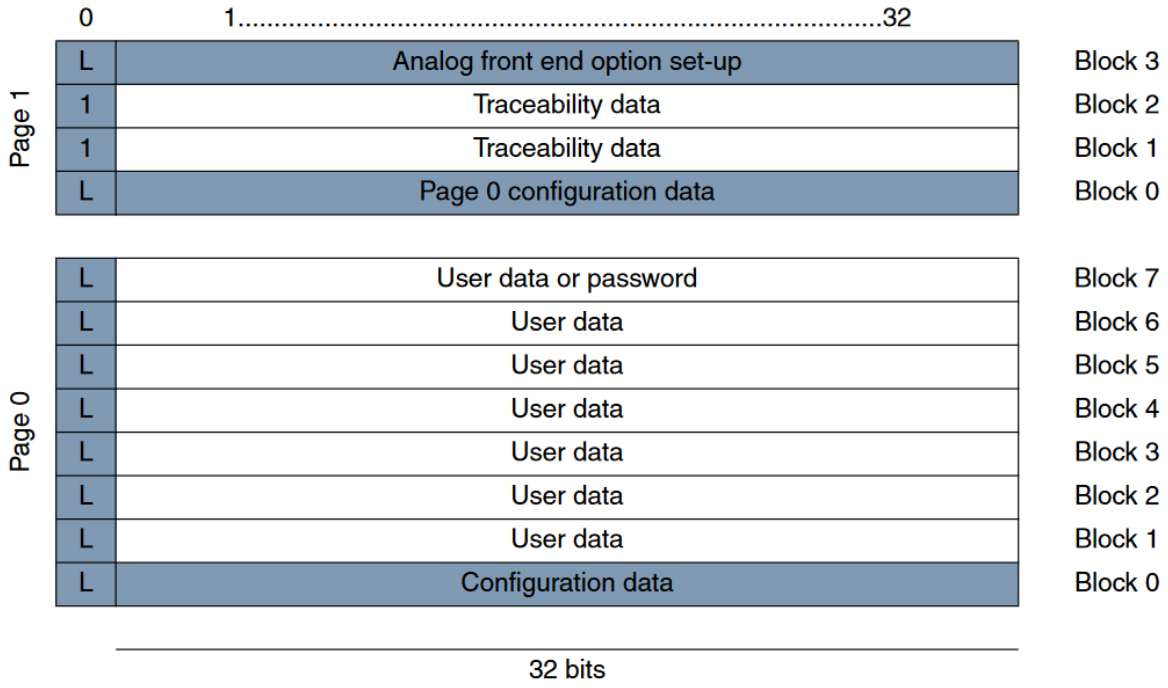

Figure 3. ATA5577 chip memory structure [23].

Only blocks 0-7 are user alterable and blocks on page 1 are locked by the manufacturer. Additionally, reading and writing are done on a block basis.

1.2.2 Communication protocol

When a tag using the T5577 transponder enters the electromagnetic field and powers up, the chip loads the information stored in block 0, shown in Figure 4.

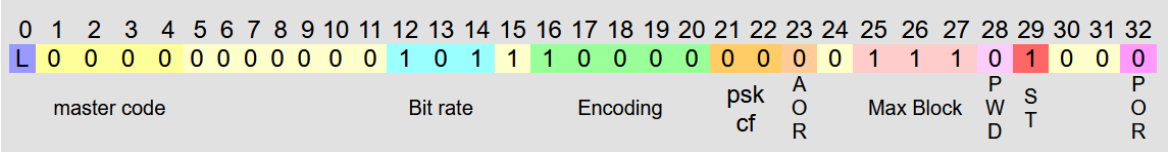

Figure 4. Block 0, configuration data of a T5577 transponder [22].

That information tells the transponder what bit rate and encoding scheme to transmit in. It then enters into Regular Read Mode, where it will start to transmit its data starting from block 1 and ending in the block number selected by the Max Block bits. After transmission of all the blocks, it will continue to retransmit the data starting from block 1 again until a command is sent to it, or it leaves the electromagnetic field. Each transmission is preceded by a special terminator sequence that allows the reader to synchronise to the first block sent [22].

1.3 Seos™

Seos™ cards operate on high frequency (13.56 MHz) and feature multiple security layers. It is currently one of the leading HID® Global products in terms of security and mobility [24].

According to a Seos™ whitepaper [3] the technology uses standards-based security instead of a proprietary security-by-obscurity approach. The standards used were developed by several companies, the academic community and government entities and are regularly checked by authorities [3]. This claim of not using a proprietary approach is debatable as the extent to which each listed standard is used, the exact memory layout and the description of the communication between the reader and the card are not publicly known. Furthermore, most of the publicly known information stems from whitepapers [3, 24] and research conducted by independent researchers.

Another Seos™ whitepaper [24] states that the technology is compatible with the ISO/IEC 7810, 7816 and 14443A standards and uses ISO/IEC 24727-3 and NIST SP800-56A aligned standards for mutual authentication between reader and card. For secure messaging, EN 14890-1 and ISO/IEC 7816 standards are used with the AES-128 encryption algorithm [24].

1.3.1 Memory

Seos™ cards come in two memory sizes: 16KB and 8KB, and the storage space used for housing the credentials is named the Seos™ vault. The vault can contain multiple digital credentials, each stored in an ADF.

Each ADF is protected with a Seos™ Authentication Key Set, which must be known to the application communicating with the card to read the digital credential. The Key Set comprises an encryption key and a MAC key which collectively establish a Secure Channel between the Seos™ vault and the reader, using the ISO 7816-4 Secure Messaging protocol [3].

The Seos™ whitepaper [3] writes that digital credentials stored in ADFs are each encapsulated in an encrypted data packet known as a Secure Identity Object (SIO). An illustration of the memory model is shown in Figure 5.

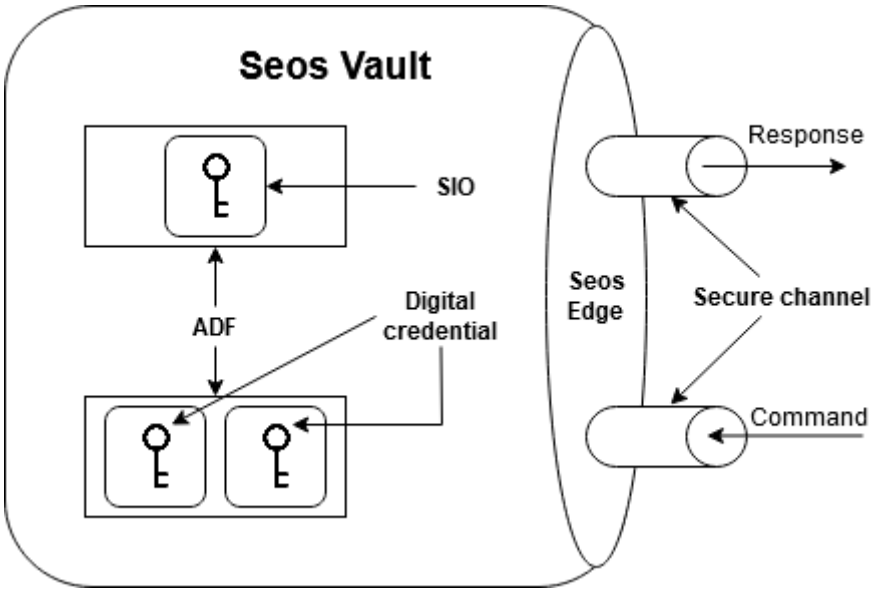

Figure 5. An illustration of the Seos™ technology memory layout.

The SIO is signed at the time of creation and this signature is validated each time a credential is used. This enables the relying system to verify that it is an authentic credential and prevents an attacker from creating a forged credential for a known User ID [3].

According to the Seos™ whitepaper [3], the vault provides an interface called the Seos™ Edge for SIO's or other data objects to be read or stored. Seos™ establishes a secure channel with the reader that is layered on top of the underlying transport protocol [3].

Since the Seos™ applet is based on the ISO/IEC 7816-4 standard its memory is also structured to use dedicated files (DFs) and elementary files (EFs) [5]. DFs host applications, groups of files or store data objects similarly to directories in Linux operating systems. EFs store data and can not be parents of other files. The standard also provides two types of file organisation layouts: a hierarchical structure with a root DF named master file (MF), seen in Figure 6, and a parallel structure without an MF or hierarchy shown in Figure 7.

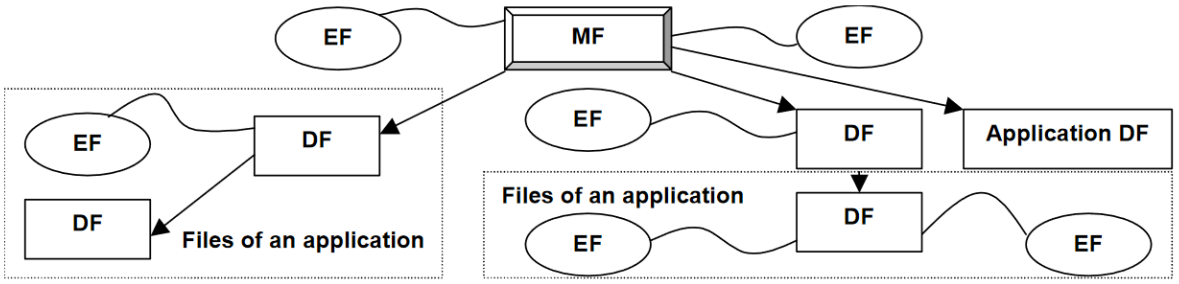

Figure 6. Example of a hierarchical memory structure [5].

Figure 7. Example of an independent application DFs structure [5].

Selecting a structure allows access to its data and in the case of DFs, access to their sub-structure. Structure selection can be made using a DF name, file identifier, path or short EF identifier [5].

1.3.2 Communication protocol

The initial dialogue between the reader (PCD) and the card (PICC), the operating frequency and other procedures in the communication layer are based on the ISO/IEC 14443 Type A standard [24]. The communication is conducted through the following consecutive operations, specified in the ISO/IEC 14443-2 standard [25]:

1. activation of the PICC by the radio frequency operating field of the PCD;
2. PICC waits silently for a command from PCD;
3. transmission of a command by PCD;
4. transmission of a response by PICC.

It should be noted that transmissions from PCD to PICC have a different encoding than transmissions from PICC to PCD. The specifics can be seen in Figure 8.

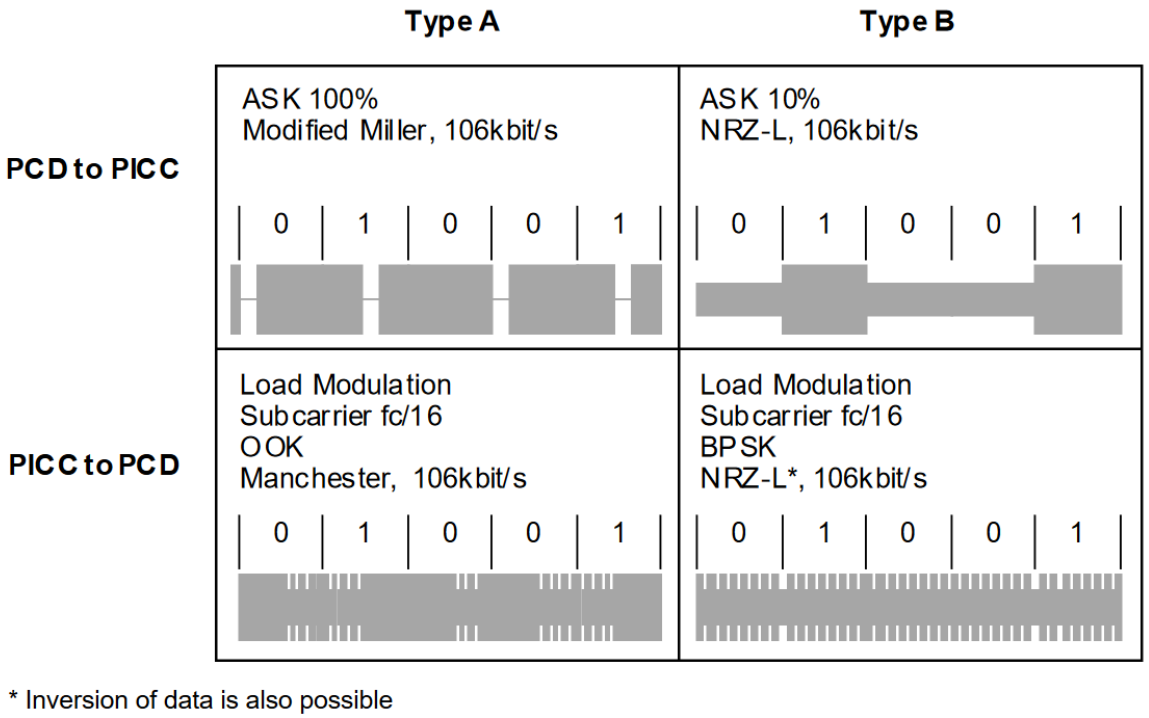

Figure 8. Data transmission specifics between PICC and PCD according to ISO/IEC 14443-2 [25].

To detect PICCs that enter its energising field, a PCD sends repeated REQUEST or WAKE-UP commands and looks for an answer to request (ATQ) response. Upon receiving an acknowledgement from the PICC, the PCD starts an anticollision loop [25]. A flowchart of the initialisation procedure is attached in Appendix 1 and a flowchart of the anticollision loop can be seen in Appendix 2.

After completing the anticollision loop, the PICC and PCD move to using commands and protocols specific to ISO/IEC 14443-4. The PCD first selects the PICC it wants to communicate with using the request for answer to select (RATS) command. After receiving an answer from the PICC, it sets up the protocol and parameters for further communication [25].

After the setup, the Seos™ card and reader use data blocks for data exchange. A diagram of the block format can be seen in Figure 9. The mandatory prologue field defines which type of block is sent, an I-block, an R-block or an S-block. I-blocks are used to convey information for the application layer and the contents of the information field are dictated by the ISO/IEC 7816-4 standard. R-blocks are used to convey positive or negative acknowledgements and S-blocks are used for exchanging control information [26].

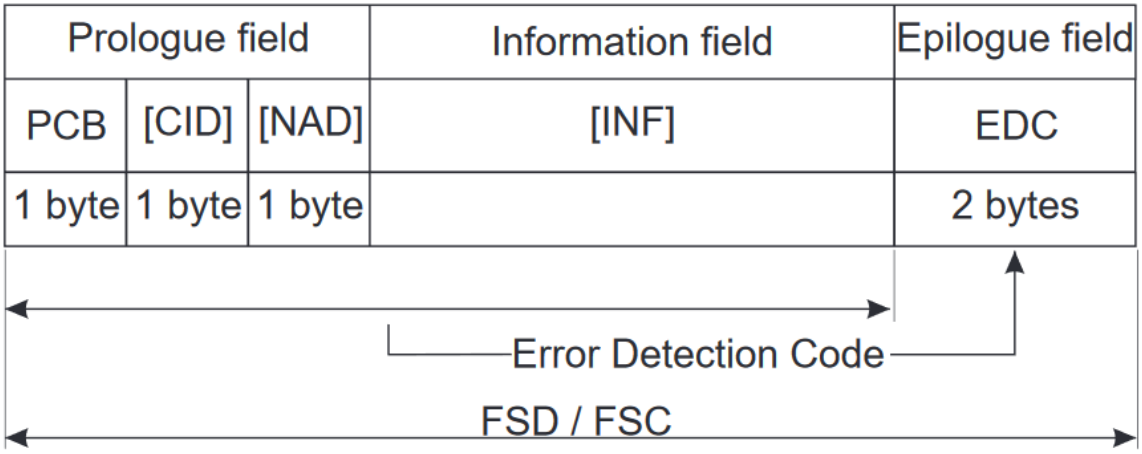

Figure 9. The byte arrangement of a data block [26].

The mandatory epilogue field contains two bytes of error detection code and is used to determine if the message was altered or corrupted during transmission.

Each I-block carries in its information field an application protocol data unit (APDU). All application interface communication is done by the reader sending an APDU command and the card responding with another APDU. These are known as command-response pairs. Table 1 shows the basic structure of an APDU and Table 2 further describes each command-response APDU field.

Table 1. Basic APDU command-response structure [5].

Command						
Header (required)				**Body (optional)**		
CLA	INS	P1	P2	Lc	Data field	Le

Response		
Data (optional)	**Trailer (required)**	
Data field	SW1	SW2

Table 2. APDU command-response pair byte arrangement [5].

Field	Description	No. of bytes
Command header	Class byte denoted CLA	1
	Instruction byte denoted INS	1
	Parameter bytes denoted P1-P2	2
Lc field	Absent for encoding Nc = 0, present for encoding Nc > 0	0, 1 or 3
Command data field	Absent if Nc = 0, present as string of Nc bytes if Nc > 0	Nc
Le field	Absent for encoding Ne = 0, present for encoding Ne > 0	0, 1, 2 or 3
Response data field	Absent if Nr = 0, present as string of Nr bytes if Nr > 0	Nr (at most Ne)
Response trailer	Status bytes denoted SW1-SW2	2

In simplified terms, an APDU is a container that holds a complete command sent to the card or a complete response from the card.

Nc, Ne and Nr denote the number of bytes contained in the command APDU's data field, expected from the card and included in the response respectively. The Lc and Le values encode the Nc and Ne values according to the ISO/IEC 7816-4 standard [5].

2. Experiments and investigation

This chapter lists all experiments and attempts that were tried to gather more information on the two proprietary technologies Indala and Seos, which are in focus of this thesis. The results of these experiments are shown in Chapter 3.

After examining the collected material detailed in the first chapter, the investigation into both technologies started with finding valid cards for each technology and buildings that used them to test how the technologies operate in practice. Afterwards, each card was analysed using the commands implemented in the RFID Research Group's (RRG) firmware[1] running on a Proxmark 3 RDV4.

2.1 Indala

To analyse the Indala technology also from the perspective of security, two in-depth inquiries were conducted: Indala UID analysis and possibility of card cloning or replay attacks.

2.1.1 UID analysis

To test if the Indala card UIDs used in the UT buildings contained any predictable pattern it was necessary to find valid Indala cards and collect their UIDs. People who had valid cards to the buildings using Indala were explained what tests would be conducted and asked to present their token. To sniff each card's UID the tag was placed on the Proxmark and the command '`lf search`' was issued to the device. This yielded a response similar to the following.

```
[usb] pm3 --> lf search
[=] NOTE: some demods output possible binary
[=] if it finds something that looks like a tag
[=] False Positives ARE possible
[=]
[=] Checking for known tags...
[=]
[+] Indala (len 224) Raw: 80000003e506e0e07d65422730a7f9ca945773bc7379fcf93ed79292
[+] Valid Indala ID found!
[+] Chipset detection: EM4x05 / EM4x69
[?] Hint: try `lf em 4x05` commands
```

[1] https://github.com/RfidResearchGroup/proxmark3 (01.05.2021)

In total, 13 different valid Indala card UIDs were gathered for nine different buildings. The collected UIDs were compared to each other using Levenstein distance. The UIDs and their analysis results are reported in the third chapter.

2.1.2 Card cloning

Suggestions of being able to clone an Indala tag onto a T5577 card could be found while exploring the commands implemented in the RRG's firmware[2]. Since valid Indala UIDs had already been collected, the next step was to determine if making a copy of the card was possible and if a clone could be used instead of the original card to gain entrance. This test was selected to see if the Indala technology was susceptible to card cloning or replay attacks.

To test this, a T5577 card was placed onto the Proxmark and the following command was typed.

```
[usb] pm3 --> lf indala clone -r
80000003e519c0e50b7542372b97b8cb945773bc7379fcf93ed79292
[=] Preparing to clone Indala 224bit to T55x7 raw
80000003E519C0E50B7542372B97B8CB945773BC7379FCF93ED79292
[+] Blk | Data
[+] ----+------------
[+] 00 | 000820E0
[+] 01 | 80000003
[+] 02 | E519C0E5
[+] 03 | 0B754237
[+] 04 | 2B97B8CB
[+] 05 | 945773BC
[+] 06 | 7379FCF9
[+] 07 | 3ED79292
[+] Data written and verified
[+] Done
```

The 56 character string in the command is an Indala UID in hexadecimal form that was previously collected.

[2] https://github.com/RfidResearchGroup/proxmark3 (01.05.2021)

2.2 Seos™

To analyse the Seos™ technology two tests were conducted: UID analysis and communication sniffing and decoding. When exploring Proxmark's commands, none were explicitly found for Seos™ cards. The '`hf 14a`' commands could be used instead since the Seos™ technology is based on the ISO/IEC 14443 type A standard [24].

2.2.1 Communication sniffing and decoding

A series of communication recordings were made to understand better how and what data is exchanged and potentially find vulnerabilities in the Seos™ technology. To sniff what was being sent from the reader to the card and vice versa, the Proxmark 3 was placed in sniffing mode between a Seos™ card and a reader. Figure 9 depicts this process.

Figure 9. Sniffing card-reader communication using the Proxmark 3.

To put the Proxmark in sniffing mode, the command '`hf 14a sniff`' was used. After placing the card on the reader once or multiple times with the Proxmark in the middle, one could list the data exchange intercepted with the '`trace list -t 14a`' command.

This process was conducted various times with multiple cards on different readers in the Delta and Pipedrive buildings, as those were the only buildings where access cards could be obtained for testing. The communication traces were in the form of hexadecimal values. These values were then translated into meaningful information using the communication standards known to be used for Seos™ technology. The results are published in the next chapter.

2.2.2 Collecting and analysing UIDs

An observation made early in the testing was that each time a Seos™ card was activated, it had a different UID value. This was also confirmed by the ISO/IEC 14443-3 standard, which states that a UID beginning with '08' is followed by three random byte values that are dynamically generated [12]. If the UID value was later in the communication process used as a nonce or seed value for the encryption algorithm, it would be important to look for patterns in the UIDs. To collect large numbers of UIDs the Proxmark command '`hf 14a cuids n`' was used, where n is the number of UIDs to be collected.

In total 93171 consecutively generated UIDs were collected and analysed. The dataset was tested for UID collision frequency and plotted on a histogram to spot any abnormalities. The results are summed up in Chapter 3.

2.3 Java card

When sniffing communication between a Seos™ card and a reader using a Proxmark the results were often corrupted and nonsensical requiring multiple attempts to record a clean dialogue. Furthermore, programming a Proxmark to act as a reader is complicated and was not attempted as it required an understanding of the system itself. A solution to this was to program a Java card instead.

A Java card is a smart card that implements the Java Card technology [27]. To program a Java card one needs to write an applet in a subset of the Java programming language and load it onto the card.

3. Results

This chapter will present what results could be found when testing and researching the Indala and Seos™ technologies. Additionally, an analysis of the created Java card applet is made.

3.1 Indala

Researching the Indala UIDs showed that each UID was 224 bits long and they all shared the same beginning 5 bytes and ending 12 bytes. Analysis of the collected UIDs in their hexadecimal format using Levenshtein distance also showed that on average all UIDs were about 12 nibbles or 6 bytes different from each other. The minimum difference between two recorded UIDs was three nibbles and the maximum was 17. Additionally, the median distance between two UIDs was 13 nibbles. This may in some cases be too small a variance to protect against brute-forcing attacks, but more research would need to be conducted to assess this properly.

A table of the collected UIDs' Levenshtein distances is appended to the thesis in Appendix 3. The script for calculating the Levenshtein distances is also included in the *Python Scripts* directory on Github[3].

There was no logical pattern that could be derived from the varying nibbles and the 224-bit format is likely not created specifically for UT but instead for an OEM. This conclusion was made from the fact that some UIDs shared on the RRG Discord server[4] also have the 224-bit format. Additionally, the 224-bit format is likely prevalent because there exists a Proxmark command for cloning Indala UIDs with the same bitlength.

Multiple different UT Indala cards were successfully tested for card cloning attacks by making working clones using a T5577 card. The tests showed that none of the marketed features of FlexSecur® prevent card cloning. The data on the card may be encrypted, but the communication can still be intercepted. A cloning device like the Proxmark is able to act as a reader, record the data sent to it and later make a clone or replay the intercepted information. Indala cards transmit the static ciphertext that is stored in their memory to the reader, which in turn decrypts it. Therefore copies of the card will be able to do the same.

The fact that T5577 cards can operate as Indala cards indicates some details about the Indala technology. It first indicates that Indala cards have about the same amount of memory that T5577 cards have. The fact that Indala UIDs are up to 224 bits long suggests that this is the

[3] https://github.com/HainLuud/SeosRecorder/tree/master/Python%20Scripts
[4] https://discord.gg/ABdEqZkJDs (01.05.2021)

minimum amount of memory on an Indala card. Furthermore, the working clones show that Indala cards operate using PSK2 modulation. Bits 16-20 of the T5577's configuration block, as shown in Figure 4, were `00010` which according to the chip's datasheet define the card's modulation to be PSK2 [23].

While recording the reader and card communication using the Proxmark, no commands sent from the reader were detected. This likely indicates that the first and second step of the communication process laid out in Figure 2 happen based on the data that the Indala card beams out when placed in the powering electromagnetic field. It also suggests that the password for authenticating an Indala card to an Indala reader is added or appended to the broadcasted data and that only the authentication data is encrypted. It also suggests that if one were to put an electromagnetic field near the card it would immediately start to broadcast its password even if not in the presence of an Indala reader.

When presented with a low-frequency non-Indala card, the UT readers did not react in any way, and when presented with an Indala card containing a nonvalid UID, the readers reacted with a short tone signal. Only when presented with an Indala card containing a valid UID does the door open and the reader reacts with a long tone signal. This behaviour suggests that the UT Indala readers may be using FlexSecur® technology.

3.2 Seos™

Because of the COVID-19 induced restrictions, the only traces that could be recorded were from two buildings in Tartu: the Delta and Pipedrive buildings, Narva mnt 4 and Paju 2, respectively. Regrettably, the recordings made at Paju 2 were significantly scrambled and only the first APDUs could be fully discerned. More facilities should have been tested to give a better generalisation of the Seos™ communication protocol.

The series of recordings and the subsequent data processing yielded a summary of the consecutive commands used in a successful communication layer interaction between a Seos™ card and reader. This summary can be seen in Table 3. A complete communication transcript of the transmitted bytes and their meanings can be viewed in Appendix 4.

Table 3. Summary of the communication layer PICC-PCD communication

Standard	Source	Command/Response
ISO/IEC 14443-3	PCD	WAKE-UP (put the PICC in the READY state)
ISO/IEC 14443-3	PICC	ATQA (acknowledge that it is in the READY state)

ISO/IEC 14443-3	PCD	ANTICOLLISION (start the anticollision loop)
ISO/IEC 14443-3	PICC	UID
ISO/IEC 14443-3	PCD	SELECT_UID
ISO/IEC 14443-3	PICC	SAK (anticollision loop ends)
ISO-IEC 14443-4	PCD	RATS
ISO-IEC 14443-4	PICC	ATS
ISO-IEC 14443-4	PCD	PPS (protocol and parameter selection)
ISO-IEC 14443-4	PICC	Acknowledge to PPS
ISO-IEC 14443-4	PCD	I-block
ISO-IEC 14443-4	PICC	I-block response

It should be noted that the UID is random each time the card is powered on. All other commands on the communication layer were constant except for the APDUs sent using I-blocks. This changing UID value initially suggested that the Seos™ technology was using the rolling code approach for authentication, but as can be seen from the intercepted communication, the technology instead uses command-response authentication.

Analysing the collected communication traces between a Seos™ card and reader based on the ISO/IEC 7816-4 standard [5], the application-layer communication can be summarised with the following five commands:

1. SELECT DF;
2. Proprietary command;
3. Start EXTERNAL AUTHENTICATION;
4. Continue EXTERNAL AUTHENTICATION;
5. GET DATA.

With the first APDU, shown in Table 4, the reader sends a SELECT command where it specifies a dedicated file (DF) by its name. The card responds with a set of file control parameters and file management data. In every intercepted trace, the first command-response pair was always the same, even for the Paju 2 recordings.

Because the first command is always a SELECT DF command, it is likely that the Seos™ technology is using the parallel independent application DFs memory structure shown in Figure 6. Furthermore, the DF that the readers first select is likely an application DF also called the Seos™ Vault in the technology's whitepapers [3].

Table 4. Example of the first command-response APDUs

Command APDU		
CLA	`00`	Plain. Only command in chain, no secure messaging (SM), using logical channel 0
INS	`a4`	SELECT command
P1-P2	`04 00`	Select the first or only occurrence of the DF by its name, return file control information
Lc	`0a`	Length of data
Data	`a0 00 00 04 40 00 01 01 00 01`	
Le	`00`	Length of expected response

Response APDU		
Data	`6f 0c 84 0a a0 00 00 04 40 00 01 01 00 01`	BER-TLV encoded file control parameters and file management data.
Status	`90 00`	Normal processing

The second command APDU, depicted in Table 5, is listed as proprietary in the ISO/IEC 7816-4 standard [5]. Additionally, the class byte is defined as representing a proprietary class. The command sent to the card sometimes varies from card to card when used on the same door, and the response from the card is always different. The command APDU recorded at Paju 2 was unlike the two variations seen in the recordings made in the Delta building. This leaves open the meaning of the second I-block. It is possible that this command selects an ADF, as was explained in Chapter 1.3.1, since there can be multiple ADFs within a Seos™ vault.

Table 5. Example of the second command-response APDUs

CLA	INS	P1-P2	Lc	Data	Le	Resp Data	Status
`80`	`a5`	`04 00`	`2a`	`06 12 2b 06 01 04 01 81 e4 38 01 01 02 01 18 01 01 91 75 05 06 14 2b 06 01 04 01 81 e4 38 01 01 02 01 18 01 01 81 23 91 26 01`	`00`	`cd 02 02 06 85 38 27 ee 1e 43 a3 b6 16 0a 00 a3 74 1a 08 51 b1 e8 7d f3 00 10 ce 97 95 82 0c a5 5f 5b 72 31 f0 92 47 e8 bf 32`	`90 00`

						52 e5 0d 9e 03 29 cf a2 b0 94 5f 0f 39 b6 d2 ab d8 5e 4e 30 8e 08 f3 12 5b 0e b8 dd 2c d8	

The odd instruction byte value indicates that the command's data field is BER-TLV encoded. Decoding it using Steven Murdoch's online TLV decoder[5] results in two universal class data objects with unknown meanings. The response APDU also contains three BER-TLV encoded data objects, two of which have unknown meanings and the last one identifying a cardholder verification method.

The third I-block command, shown in Table 6, is again the same throughout the traces and it starts the GENERAL AUTHENTICATION procedure using the EXTERNAL AUTHENTICATION function. According to the ISO/IEC 7816-4 standard [5] in this authentication procedure, an entity in the card authenticates an entity in the outside world. Therefore the command sent to the card communicates the start of the authentication procedure and issues a challenge request to the card. The card replies with BER-TLV encoded data and each observed response has been different.

Table 6. Example of the third command-response APDUs

CLA	INS	P1-P2	Lc	Data	Le	Resp Data	Status
00	87	00 01	04	7c 02 81 00	00	7c 0a 81 08 f5 48 90 3c cb 4d dc 9d	90 00

In the fourth I-block, the authentication process continues as shown in Table 7. In these command-response APDUs the reader issues a response and the card verifies it. Strangely the ISO/IEC 7816-4 [5] standard states that the response data field is absent following the reader's request, but this did not align with the recordings. This behavior of including a response data field matches the MUTUAL AUTHENTICATION function but the recordings then lack the first 'Witness' command-response part of the authentication procedure. The data fields of both the command and response were unique throughout the recordings but never empty. The `dumpasn1` Linux command line tool[6] decoded both command and response data field BER-TLV values as APPLICATION objects.

[5] https://emvlab.org/tlvutils/ (01.05.2021)
[6] http://manpages.ubuntu.com/manpages/precise/man1/dumpasn1.1.html (01.05.2021)

Table 7. Example of the fourth command-response APDUs

CLA	INS	P1-P2	Lc	Data	Le	Resp Data	Status
00	87	00 01	2c	7c 2a 82 28 0e 0e 01 fa 44 16 73 23 84 0f 09 8a d5 19 36 08 cb 56 6b 61 8c f8 e3 d2 f3 dd 49 3a ed ae c1 88 b7 5d b4 40 17 c2 26 4b	00	7c 2a 82 28 93 a3 1e 29 cb 27 5d e3 d1 9e d4 fd 4d 40 14 f6 6e 67 15 e1 c7 33 30 6d 9a cb 20 7b eb bc f6 06 38 38 c8 e9 32 10 d7 db	90 00

The fifth and last I-block command, shown in Table 8, is GET DATA, which asks the card to send the content of an elementary file (EF) to the reader over a secure messaging channel. The card responds with a plain value encoded in BER-TLV but sent over an encrypted channel. According to Steven Murdoch's TLV decoder[7], the command's data field contains an unknown object, a transaction certificate object list and a cardholder verification method list. The response also contains an unknown object and a cardholder verification method list, but also a transaction personal identification number data. This is likely the point where the reader retrieves the SIO data from the card.

Table 8. Example of the fifth command-response APDUs

CLA	INS	P1-P2	Lc	Data	Le	Resp Data	Status
0c	cb	3f ff	16	85 08 8b a2 07 07 4e 23 f9 d5 97 00 8e 08 60 d4 ff 77 3b 58 6d 53	00	85 40 63 f2 fc 76 63 31 31 90 46 92 c4 a9 bd d3 3d 93 b7 f3 8a ff b1 c4 f3 c8 fa e7 4a 0f 38 21 7a 30 80 f2 ed 58 54 fb 47 fb d0 c4 65 39 27 2a 7b 62 89 61 bb 0d a8 26 6f ce 87 ab f2 19 e3 38 5f 1c 99 02 90 00 8e 08 50 ef fd 06 b8 9a b3 3c	90 00

[7] https://emvlab.org/tlvutils/ (01.05.2021)

The analysis of the 93,171 collected Seos™ card UIDs could not determine any predictable patterns. There were many collisions between the UIDs which were initially thought to be due to the similarity of the situation to the birthday paradox known in probability theory. The paradox highlights that it takes only 23 random people to have around a 50% likelihood of two of them sharing the same birthday [28].

Since each UID was 4 bytes long and shared the first byte, there are 16,777,216 possible unique UIDs. If one assumed that choosing random UIDs is similar to the birthday paradox, one could calculate how many random values need to be generated before the probability of two or more of them being the same exceeds 50%. This can be calculated using the formula

$$1 - P(n) \geq 0.5$$

where P(n) is the probability of randomly choosing n unique UIDs. This 50% threshold is exceeded with 4,823 random UIDs, significantly less than the number of values collected. This means that there almost certainly had to be UID collisions in the collected dataset.

When counted, there were 16,802 unique UID values in the Seos™ generated dataset, of which 9,338 occurred only once and the other 7,464 repeated in total 83,833 times. In comparison, a test was conducted where the same amount of UIDs were generated using Python's Numpy library[8] as collected using the Seos™ card. As a result of running this test 1000 times there were on average only 257 collisions.

Additionally, random samples of 500 UIDs were picked from the collected dataset and also generated using the Numpy library. On average there were around 20 collisions in the dataset which had randomly picked Seos™ generated UIDs. However after generating 500 UID samples over one million iterations, none had as many or more collisions as the Seos™ dataset average. This suggests that it is statistically highly unlikely that the Seos™ generated UIDs are random, as the amount of collisions observed is much higher than what can reasonably be expected from a random dataset. It may also indicate that the random number generator (RNG) in the Seos™ cards is weak and that with further analysis the generated UIDs or the card's encryption could be predicted. This randomness testing was done on different hardware platforms which may have produced slightly different results than if the comparison measurements were done on perhaps a Java card platform. Further testing should be done to get a more accurate indication of how random the Seos™ card's RNG is.

[8] https://numpy.org/ (01.05.2021)

Plotting the collisions of each UID in Figure 10 also shows that the amount of collisions per UID is not distributed evenly.

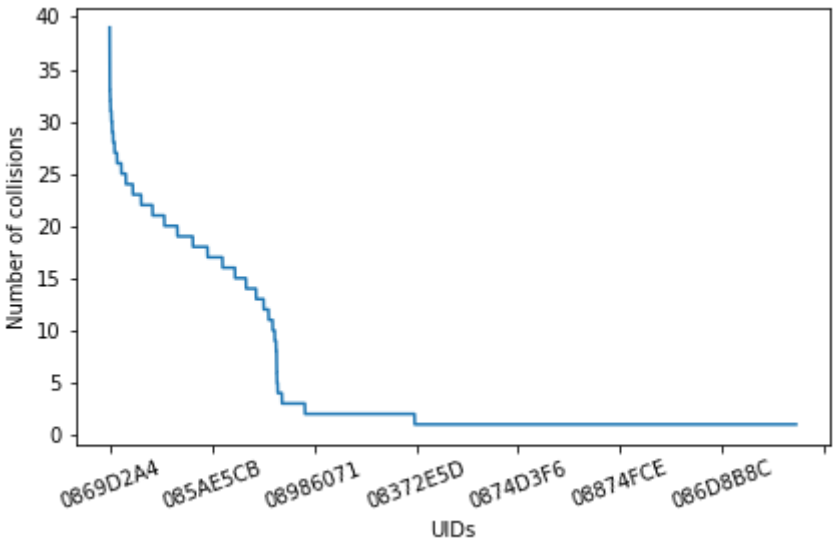

Figure 10. Number of collisions per UID.

However, there did not appear to be any specific range where the UIDs were significantly more frequent, as shown in Figure 11.

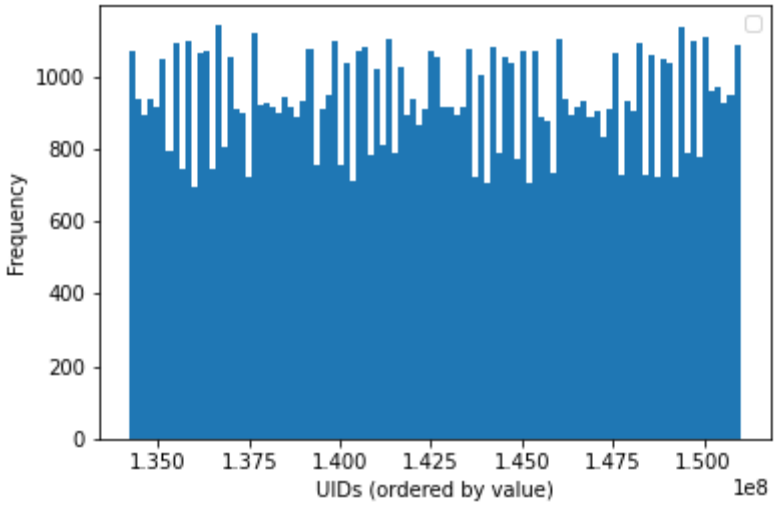

Figure 11. Histogram of the collected UIDs using 100 bins.

The results and the code used for running the tests on UIDs are uploaded in the SeosRecorder Github repository[9].

Likewise, with the Seos™ UIDs, an attempt was made to sniff enough traffic between cards and readers to see if any of the protocol traces were identical to another. Regrettably, no

[9] https://github.com/HainLuud/SeosRecorder/tree/master/Seos%20UIDs

convenient way to amass a large number of clear and unscrambled samples was found. Of the small number of samples that were recorded, none were identical. Additionally, a large amount of the recordings were garbled.

3.3 Java card applet

The created Java card applet[10] can successfully record the commands a reader sends and is able to respond to them. Once a Java card using the SeosRecorder applet is placed onto a ISO/IEC 14443 Type A standard compatible reader, the card communicates with it using pre-programmed responses and records every command it receives. The memory is currently 1536 bytes which can store around 7 full communication recordings. Because of the limiting factor of not being able to use larger than 256-byte arrays it is difficult to increase the card's storage capacity for recordings.

In addition to the development of the Java card applet two Python scripts were developed, one for transmitting a complete set of commands collected while intercepting dialogue between a real reader and Seos™ card, and one for retrieving the applet's memory content.

To differentiate between commands recorded in memory, each command is appended with three bytes of 0xFF. This separator was chosen because according to the ISO/IEC 7816-4 standard [5] three 0xFF values should not occur consecutively after a command's data field.

The recording retrieval script sends memory retrieval commands to the applet which in turn responds to each with the corresponding memory array. Since larger than 256 byte arrays are not possible to construct in the Java subset language for Java cards, this method of writing a command for each memory array was selected. The script then parses the received memory arrays and outputs the APDU structure of each command the card received.

The applet does at times record anomalous SELECT commands that were not explicitly sent using the Python scripts included in the project's repository[11]. This may be due to the ACR1252U USB NFC reader, that was used for testing, sending some information in the background. Another potential issue with the SeosRecorder applet is that its UID and ATR values can not be changed. Since the ATR value is used in other contexts to distinguish a card's technology then having a value other than what Seos™ has, may yield false results

[10] https://github.com/HainLuud/SeosRecorder
[11] https://github.com/HainLuud/SeosRecorder/tree/master/Python%20Scripts

during testing. Additionally, its UID value is 7 bytes long, which is also different from Seos' 4 byte UID.

A Java card is less conspicuous to use than a Proxmark, but in general, the card is not a tool that could replace it. Unlike the Proxmark, the card can not intercept the communication between a Seos™ card and a reader since it can only record the commands if it is itself selected by the reader. The applet is still useful for conducting further research into high-frequency cards similar to Seos™ as it is more accurate in collecting the commands sent by the reader and more straightforward to program than the Proxmark.

4. Conclusions

This thesis set out to compile a more detailed and technical description of how Indala and Seos™ technologies operate. In addition to giving an overview of the technologies based on the standards and white papers currently publicly available, several measurements and analyses were conducted to inspect their communication procedure and assess them from a security perspective. Furthermore, a tool was developed to aid in researching the Seos™ technology communication protocol.

The technical overview of three RFID technologies was concluded in Chapter 1: the two most used in the University of Tartu and a third technology that was used for testing purposes. The research identified that the technologies in the focus of this thesis use very different methods for card authentication, have contrasting memory layouts and operate on two distinct frequencies.

The experiments conducted in Chapter 2 also brought out more details of the two technologies. The first significant finding was that despite the security measures used in the Indala technology, the cards are still susceptible to cloning and replay attacks. T5577 cards were successfully used to clone Indala cards which also gave further information on the Indala technology. Additionally, it was found that the average Levenshtein distance of the 13 Indala UIDs collected for research was 12 nibbles. The minimum difference between two recorded UIDs was three nibbles and the maximum was 17.

The second important finding was that the RNG used in Seos™ cards might be weak. In an experiment, over 93 thousand UIDs were generated using the card and then analysed. Compared to a dataset with identical size and UID format but generated using the Python Numpy library, the Seos™ dataset exhibited a statistically unlikely amount of UID collisions. It is yet unclear if and how this impacts the overall security of the technology.

Additionally, a full transcript of the Seos™ card and reader communication, shown in Appendix 4, was collected and partially decoded. The meaning of some commands and data field bytes could not be interpreted since they were either encrypted or listed as proprietary in the corresponding standard.

The third most significant outcome of this research was the development of the SeosRecorder Java card applet. The applet can help conduct further research into high-frequency cards similar to the Seos™ card as it can accurately collect commands sent by the reader. However,

the created applet may have some disadvantages stemming from the lack of possibility to change its UID or ATR values.

Further research should be conducted into the Seos™ technology as there still remain several questions to be answered regarding its use of an RNG and the values that are communicated with the reader.

5. References

[1] ISO - the International Organisation for Standardisation. ISO/IEC 14443-1:1997, http://sweet.ua.pt/andre.zuquete/Aulas/IRFID/11-12/docs/ISO-IEC_14443-1.pdf (29.04.2021)

[2] ISO - the International Organisation for Standardisation. ISO/IEC 7810:2003, https://docs.google.com/file/d/0B5vp4lGahnJFZUtUc0Y1VDg0Tzg/view (29.04.2021)

[3] iCLASS Seos: Introducing the HID Global iCLASS Seos Card, 2014. http://www.emacs.es/downloads/WP/20140723_iCLASS_Seos_Card_Whitepaper_EXTERNAL_v1.0.pdf (04.12.2020)

[4] Wehr J. Understanding anticollision: Processing multiple cards at the same time, 2003, https://www.secureidnews.com/news-item/understanding-anticollision-processing-multiple-cards-at-the-same-time/ (08.03.2021)

[5] ISO - the International Organisation for Standardisation. ISO/IEC 7816-4:2005

[6] Xiao Q. *et al.* RFID Technology, Security Vulnerabilities, and Countermeasures. *Supply Chain, The Way to Flat Organisation*, Huo Yanfang *et al.* Vienna, 2008, 404 https://cdn.intechopen.com/pdfs/6177.pdf (07.01.2021)

[7] The Levenshtein Algorithm, https://www.cuelogic.com/blog/the-levenshtein-algorithm (08.03.2021)

[8] Security Encyclopedia - Nonce, https://www.hypr.com/nonce/ (08.03.2021)

[9] Collins English Dictionary - Complete & Unabridged 2012 Digital Edition, https://www.dictionary.com/browse/rfid

[10] Rankl W., Effing W. Smart Card Handbook: Fourth Edition, John Wiley & Sons Ltd, 2010

[11] Nilsson, S. What's a seed in a random number generator? https://yourbasic.org/algorithms/random-number-generator-seed/ (08.03.2021)

[12] ISO - the International Organisation for Standardisation. ISO/IEC 14443-3:1999

[13] Keycards for Access Control Systems. https://www.getkisi.com/keycard-access-systems (26.01.2021)

[14] Norman T. Electronic Access Control, Elsevier Inc., 2012 https://dl.acm.org/doi/pdf/10.5555/2597840 (26.01.2021)

[15] Farooq U. *et al.* RFID Based Security and Access Control System. *International Journal of Engineering and Technology*, vol. 6, no. 4, 2014, 309–314 http://www.ijetch.org/papers/718-B10136.pdf (04.12.2020)

[16] A Guide to RFID Types and How They Are Used. https://www.atlasrfidstore.com/a-guide-to-rfid-types-and-how-they-are-used/ (04.12.2020)

[17] RFID Security & Privacy.

https://www.electronics-notes.com/articles/connectivity/rfid-radio-frequency-identification/security-privacy.php (04.12.2020)

[18] HID® Indala® FlexKey® Keytag

https://www.hidglobal.com/products/cards-and-credentials/indala/flexkey-keytag (27.01.2021)

[19] HID® Indala® FlexSecur® Technology.

https://www.hidglobal.com/sites/default/files/indala-flexsecur-wp-en.pdf (04.12.2020)

[20] Indala® Prox 125 kHz Proximity – Credentials. https://www.hidglobal.com/system/files/doc_eol_expired_files/indala-prox-cards-ch-en.pdf (04.12.2020)

[21] Leavitt A. What Is The Wiegand Protocol? 2018, https://getsafeandsound.com/2018/09/wiegand-protocol/ (29.01.2021)

[22] T5557 Read/Write RFID Transponder https://www.priority1design.com.au/t5557_rfid_transponder.html (04.12.2020)

[23] Microchip Technology. ATA5577C

http://ww1.microchip.com/downloads/en/DeviceDoc/ATA5577C-Read-Write-LF-RFID-IDIC-100-to-150-kHz-Data-Sheet-DS70005357B.pdf (30.01.2021)

[24] Seos™ Card.

https://www.hidglobal.com/sites/default/files/resource_files/pacs-seos-card-ds-en_0.pdf (29.04.2021)

[25] ISO - the International Organisation for Standardisation. ISO/IEC 14443-2:1999

[26] ISO - the International Organisation for Standardisation. ISO/IEC 14443-4:2000

[27] Development Kit User Guide - Preface. https://docs.oracle.com/en/java/javacard/3.1/guide/preface.html#GUID-07C116D2-4E9D-43CB-99CF-D79DC46C077A (08.03.2021)

[28] Understanding the Birthday Paradox.

https://betterexplained.com/articles/understanding-the-birthday-paradox/ (28.03.2021)

Appendix 1

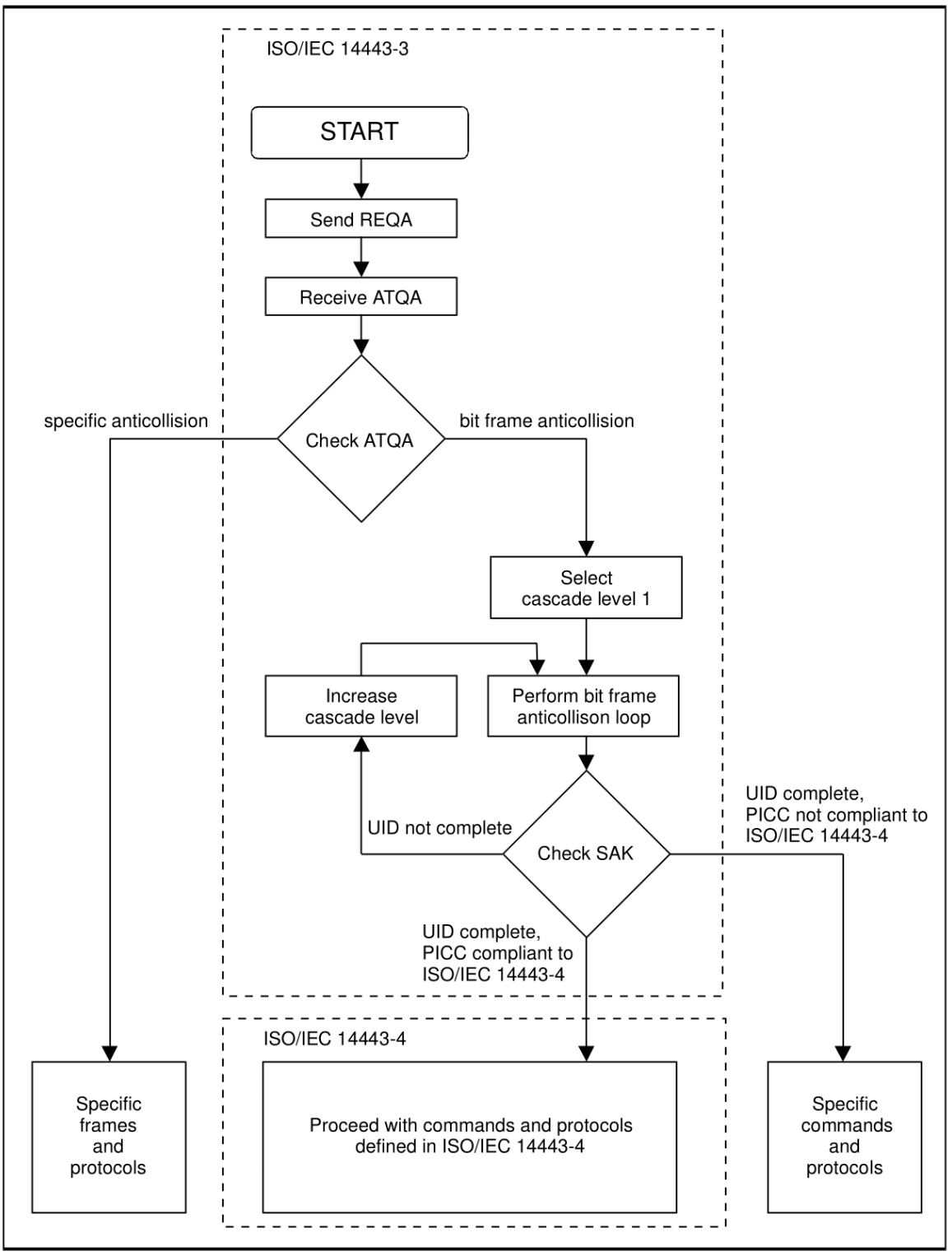

Figure 12. Smart card initialisation procedure flowchart [12].

Appendix 2

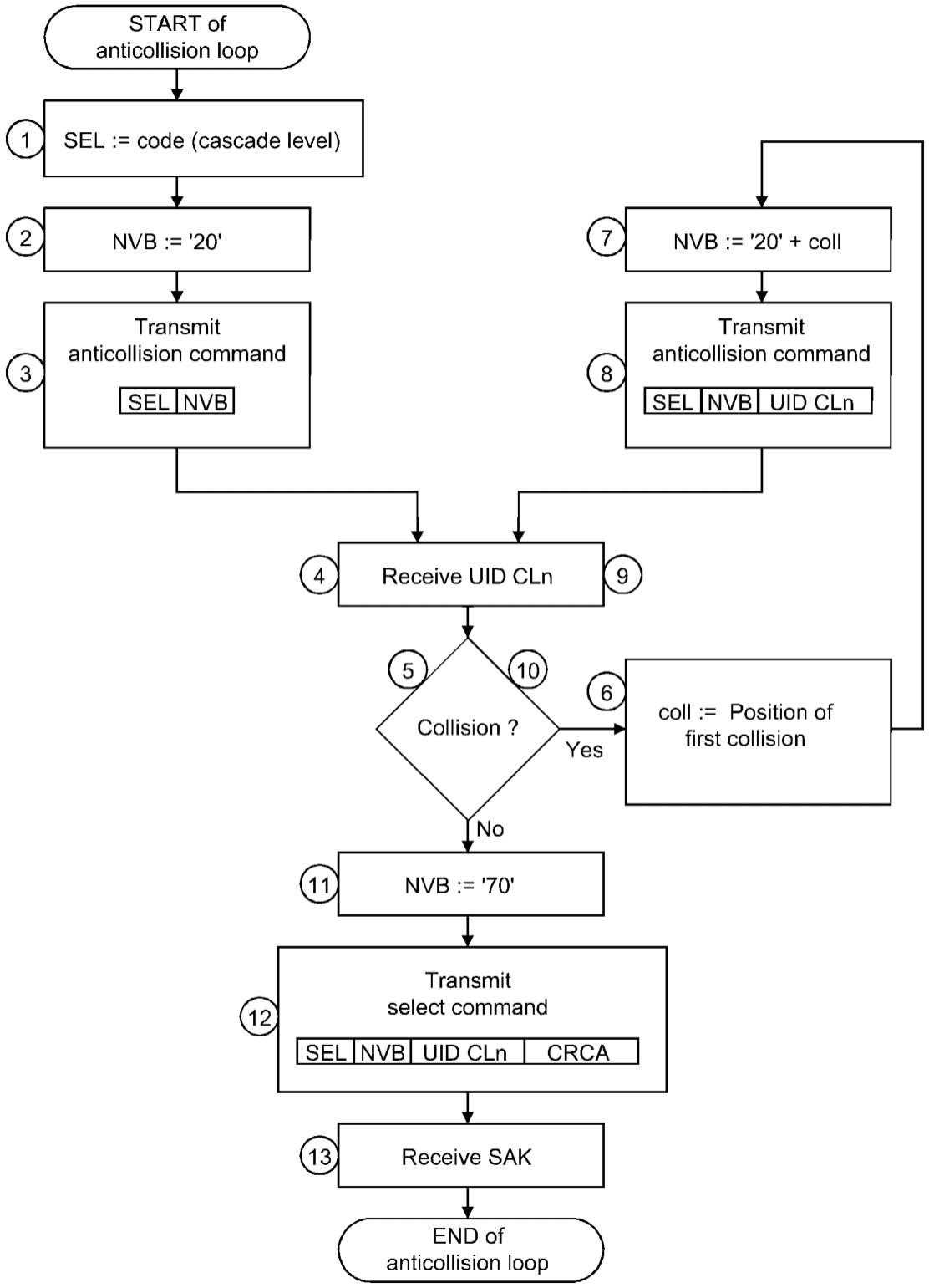

Figure 13. Anticollision loop flowchart [12].

Appendix 3

Table 8. Levenshtein distance of the collected Indala UIDs

	UID 1	**UID 2**	**UID 3**	**UID 4**	**UID 5**	**UID 6**	**UID 7**	**UID 8**	**UID 9**	**UID 10**	**UID 11**	**UID 12**	**UID 13**
UID 1	0	7	3	13	13	13	14	10	9	17	13	14	14
UID 2	7	0	7	11	12	12	11	10	10	14	13	12	12
UID 3	3	7	0	14	13	13	15	9	10	17	12	14	14
UID 4	13	11	14	0	11	14	9	13	12	11	11	13	13
UID 5	13	12	13	11	0	14	4	12	15	13	13	13	13
UID 6	13	12	13	14	14	0	13	13	14	11	14	12	12
UID 7	14	11	15	9	4	13	0	14	13	12	14	14	13
UID 8	10	10	9	13	12	13	14	0	7	13	12	13	15
UID 9	9	10	10	12	15	14	13	7	0	13	12	15	13
UID 10	17	14	17	11	13	11	12	13	13	0	11	12	12
UID 11	13	13	12	11	13	14	14	12	12	11	0	6	8
UID 12	14	12	14	13	13	12	14	13	15	12	6	0	5
UID 13	14	12	14	13	13	12	13	15	13	12	8	5	0

Appendix 4

Table 9. Full Seos™ card and reader communication transcript.

Direction	Data	Explanation
Rdr > Tag	52	WUPA
Tag > Rdr	01 00	Ack WUPA
Rdr > Tag	93 20	Reader starts anticollision loop
Tag > Rdr	08 3a 74 b9 ff	Card send its random UID
Rdr > Tag	93 70 08 3a 74 b9 ff 57 12	Reader selects card by UID
Tag > Rdr	20 fc 70	Card acknowledges selection (SAK) and anticollision loop ends
Rdr > Tag	e0 80 31 73	RATS
Tag > Rdr	05 78 77 80 02 9c 3a	ATS
Rdr > Tag	d0 11 00 52 a6	PPS
Tag > Rdr	d0 73 87	Acknowledge PPS
Rdr > Tag	0a 00 00 a4 04 00 0a a0 00 00 04 40 00 01 01 00 01 00 6a 2c	1st I-block command. APDU command “SELECT by dedicated file name”.
Tag > Rdr	0a 00 6f 0c 84 0a a0 00 00 04 40 00 01 01 00 01 90 00 6f a4	1st I-block response. Returns file control information and file management data.
Rdr > Tag	0b 00 80 a5 04 00 2a 06 12 2b 06 01 04 01 81 e4 38 01 01 02 01 18 01 01 91 75 05 06 14 2b 06 01 04 01 81 e4 38 01 01 02 01 18 01 01 81 23 91 26 01 00 1c ef	2nd I-block. Proprietary class and unknown APDU command. Data field holds two unknown BER-TLV encoded data objects. Command length may vary.
Tag > Rdr	0b 00 cd 02 02 06 85 38 27 ee 1e 43 a3 b6 16 0a 00 a3 74 1a 08 51 b1 e8 7d f3 00 10 ce 97 95 82 0c a5 5f 5b 72 31 f0 92 47 e8 bf 32 52 e5 0d 9e 03 29 cf a2 b0 94 5f 0f 39 b6 d2 ab d8 5e 4e 30 8e 08 f3 12 5b 0e b8 dd 2c d8 90 00 f3 ce	2nd I-block. Response data field holds three BER-TLV objects: two unknown objects and a cardholder verification method list. Data is different each recording.
Rdr > Tag	0a 00 00 87 00 01 04 7c 02 81 00 00 9c 8d	3rd I-block. Start EXTERNAL AUTHENTICATE function

Tag > Rdr	0a 00 7c 0a 81 08 f5 48 90 3c cb 4d dc 9d 90 00 37 78	3rd I-block. Respond with BER-TLV encoded data which is different each recording.
Rdr > Tag	0b 00 00 87 00 01 2c 7c 2a 82 28 0e 0e 01 fa 44 16 73 23 84 0f 09 8a d5 19 36 08 cb 56 6b 61 8c f8 e3 d2 f3 dd 49 3a ed ae c1 88 b7 5d b4 40 17 c2 26 4b 00 a9 a5	4th I-block. Continue EXTERNAL AUTHENTICATE. Command data field contains a BER-TLV encoded APPLICATION object and is different for each recording.
Tag > Rdr	0b 00 7c 2a 82 28 93 a3 1e 29 cb 27 5d e3 d1 9e d4 fd 4d 40 14 f6 6e 67 15 e1 c7 33 30 6d 9a cb 20 7b eb bc f6 06 38 38 c8 e9 32 10 d7 db 90 00 de a3	4th I-block. Response data contains a BER-TLV encoded APPLICATION object. The object’s bytes are different each recording
Rdr > Tag	0a 00 0c cb 3f ff 16 85 08 8b a2 07 07 4e 23 f9 d5 97 00 8e 08 60 d4 ff 77 3b 58 6d 53 00 d0 0d	5th I-block. APDU command “GET DATA” asks the card to send the content of a file over a secure messaging channel. Command datafield contains a BER-TLV encoded unknown object and a transaction certificate object and cardholder verification method lists. The data field bytes are different each recording.
Tag > Rdr	0a 00 85 40 63 f2 fc 76 63 31 31 90 46 92 c4 a9 bd d3 3d 93 b7 f3 8a ff b1 c4 f3 c8 fa e7 4a 0f 38 21 7a 30 80 f2 ed 58 54 fb 47 fb d0 c4 65 39 27 2a 7b 62 89 61 bb 0d a8 26 6f ce 87 ab f2 19 e3 38 5f 1c 99 02 90 00 8e 08 50 ef fd 06 b8 9a b3 3c 90 00 cf 8b	5th I-block. The response contains a BER-TLV encoded unknown object, a cardholder verification method list and a transaction personal identification number data. The data is different each recording.
Rdr > Tag	ca 00 7a 29	End communication
Tag > Rdr	ca 00 7a 29	End communication

License

Non-exclusive licence to reproduce thesis and make thesis public

I, Hain Luud,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, An analysis of the HID® Indala and Seos™ protocols, supervised by Danielle Morgan.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Hain Luud

07/05/2021