TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technology
Department of Software Science

ITC70LT
Mikk Romulus 163296IVCM

SECURITY TESTING ESTONIAN CONTACTLESS BANK CARDS

Master's thesis

Supervisor: Professor Olaf Maennel, TalTech
Co-Supervisor: Tiit Hallas, LHV

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

ITC70LT
Mikk Romulus 163296IVCM

EESTI VIIPEMAKSEKAARTIDE TURVATESTIMINE

Magistritöö

Juhendaja: Professor Olaf Maennel, TalTech

Kaasjuhendaja: Tiit Hallas, LHV

Tallinn 2019

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Mikk Romulus

January 6, 2019

Autorideklaratsioon

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Autor: Mikk Romulus

6. jaanuar 2019. a.

Abstract

The following thesis discusses the security of EMV Contactless bank cards. EMV is an international standard for bank cards and dominates the market in most regions, any security issues with the standards or cards will have a large impact. This thesis focuses on contactless aspect of the bank cards — mode where transactions can be made without physical contact with payment terminal. As increasing number people choose convenient electronic payments over cash, their finances may be endangered by as-yet-unknown vulnerabilities. Existing research has uncovered few attack vectors, but no systematic means for security testing EMV cards has been developed. One of the goals for this thesis is to create a method to map and fuzz test the functionality of bank cards, the second is testing the security of Estonian bank cards using previously discovered vulnerabilities and the aforementioned method. Methodology created consists of mapping the available command space from fixed start condition, determining any valid parameters and payload lengths for discovered instructions and launching suitable fuzzing mode on selected commands. The results indicate that some of the test cards have potential security issues, such as possibility for relay attacks or pre-play attacks using the magnetic stripe mode payments. Fuzz testing likewise revealed irregularities in tested cards, including permanently altering the behaviour of one test card. The tools developed alongside this thesis have been made open source and are available online.

The thesis is in English and contains 70 pages of text, 5 chapters, 2 figures, 12 tables.

Annotatsioon

Käesolev lõputöö uurib *EMV Contactless* tüüpi pangakaartide turvalisust. Antud kaarte (*contactless*) nimetatakse ka viipemakse kaartideks. Nende kasutamiseks ei ole tarvis kaarti füüsiliselt lugejasse sisestada ja piisab makseterminali lähedal viipamisest. Eestis ja paljudes teistes riikides kasutatavaid nii viipe kui harilikke pangakaarte nimetatakse EMV kaartideks samanimelise rahvusvahelise pangakaartide standardi järgi. Nimetus EMV tuleneb standardi loonud organisatsioonide nimede esitähtedest — Europay, MasterCard ning Visa. Laia leviku ja töödeldavate raha mahtude tõttu on EMV kaartidega seotud turvaprobleemidel väga suur mõju. *EMV Contactless* turvalisust on ka varem uuritud ja avastatud turvanõrkusi, kuid seni pole välja töötatud süstemaatilist meetodit kaartide uurimiseks. Lõputöö üks eesmärkidest on meetodi, mis kaardistaks ja testiks (kasutades *fuzz testing* põhimõtteid) kaartide funktsionaalsust, välja töötamine. Teiseks eesmärgiks on testida Eestis väljastatavate viipemakse kaartide turvalisust, proovides kasutada nii varasemalt avastatud nõrkusi kui eelmainitud meetodiga leitud probleeme. Käesolev töö piirdub kaartide testimisega omaette, testimise alla ei kuulu makseterminalid ega muud osad maksete taristust.

Töös kasutatav metodoloogia sisaldab mitut etappi: lähtudes kindlaksmääratud kaardi algseisunditest kättesaadavate käskude kaardistamine, leitud käskudel toimivate parameetrite tuvastamine, sobiva andmevälja pikkuse leidmine, leitud käskudest valimikul *fuzz* testimine käsu iseloomule vastavas režiimis. Kirjeldatud etappide läbi viimiseks ei olnud varem sobivaid tööriistu, mistõttu arendati lõputöö käigus välja sobiv tarkvara nimetatud sammude teostamiseks. Loodud tarkvara on avatud lähtekoodiga ja Internetist alla laetav.

Töö tulemused näitavad, et testitud kaartidel on turvapuudusi, näiteks eelarvutus rünnak (*pre-play attack*) magnetriba režiimis. Potentsiaalse probleemina tuli välja teaterünnak (*relay attack*), mille testimine jäi väljapoole lõputööle ette nähtud skoopi. *Fuzz* testimise kaudu tuvastati ka ebakorrapärasusi kaartide toimimises, mille seas oli ka ühe testitud kaardi käitumise püsiv muutus. Leitud probleem ei ole tõsine ja testitud kaart toimib jätkuvalt maksevahendina, aga on sellegipoolest indikatiivne võimalike vigade suhtes.

Lõputöö on kirjutatud inglise keeles ja koosneb 70 leheküljest, 5 peatükist, 2 joonisest ning 12 tabelist.

List of abbreviations and terms

APDU Application protocol data unit is a unit of communication between smart card and reader. 8, 18, 37, 38, 55

cAPDU Command APDU is an APDU sent to the card. 8, 18

EMV EMV refers to:
1 — international industry standards for bank cards, named after the founding organisations Europay, MasterCard and Visa
2 — bank cards that use chips and asymmetric cryptography instead of magnetic stripes. 15–18, 21–24, 26–32, 34–41, 44–46, 48–51, 53, 59, 60

GPO Get Processing Options is a command sent to bank cards which is used to retrieve information necessary for finishing a transaction. 12, 22, 32, 33, 37, 38, 45–47, 50, 55

PoS Point of sale terminal is a card reader device installed at a merchant, which processes card payments. 16, 18, 22, 24, 26–29, 36, 57, 60

rAPDU Response APDU is an APDU sent from the card as reply to a command APDU (cAPDU). 18, 19

Acronyms

AFL Application file locator. 13, 22, 38, 47, 55, 57

AID Application identifier. 22

AIP Application interchange profile. 22, 28, 57

ATC Application Transaction Counter. 27, 28, 38, 40, 46, 47

ATM Automatic Teller Machine. 18, 26, 60

CDA Combined data authentication. 22

CDOL Card Risk management DOL. 22, 38, 40, 55

CNP Card Not Present. 26, 36, 44

CPLC Card Production Life Cycle. 46

CRC Cyclic Redundancy Check. 33

CVM Customer Verification Method. 22

DDA Dynamic data authentication. 22, 24, 50

ICC Intergrated Circuit Card. 13, 22, 29, 43, 44

NFC Near Field Communication. 23, 36, 44

PAN Primary account number. 22

PC/SC Personal Computer/Smart Card. 34

PEM Privacy Enhnanced Mail. 43

PIN Personal Identification Number. 14, 22, 27, 29, 36, 38, 46, 53, 60

PKI Public Key Infrastructure. 29, 53

PPSE Proximity Payment System Environment. 22

RFID Radio Frequency Identification. 23, 26, 27, 33

RFU Reserved for future use. 19

ROCA Return of Coppersmith Attack. 29, 35, 44

SDA Static data authentication. 22, 24

SFI Shoft File Identifier. 38, 44, 47, 55

TLV Tag-Length-Value. 22, 38

UN Unpredictable Number. 27, 57, 58

Contents

List of Figures

List of Tables

1. Introduction

In modern digital societies, electronic banking is as ubiquitous as personal computing or television and necessary part of modern economies. For consumers, one of the cornerstones of the banking system is the personal bank card. As technology has evolved, so has the bank card — from using magnetic stripes to chips technology to the latest innovation — contactless (swiping) payments. With contactless payment, the cardholder completes a transaction by holding their card in close proximity to the payment terminal. This procedure takes less than a second and is significantly quicker than older technology since no further verification (such as signature or Personal Identification Number (PIN) entry) is required. This method provides convenience and speed to the customer while reduced friction in completing the transaction also helps the merchants with increased spending [1]. These factors contribute to the large gain in market share for contactless payments – by September 2017 17% of bank cards and 85% of payment terminals in Estonia supported swiping payments. [2, p. 26]

In some advanced digital societies, bank cards are even starting to replace cash — in Sweden only 1% of the value of all payments in 2016 was made with cash, compared to approximately 7% in the EU and the US [3]. Estonia is also trending towards cashless future — in 2016, 327 cashless payments per capita were conducted [2, p. 7], following the UK, Denmark, the Netherlands, Finland and Sweden. As increasing numbers of people start to rely on alternatives to cash for conducting their finances, the security and trustworthiness of electronic banking, including bank cards, becomes of critical importance. Some countries, such as the US [4], the UK [5], Australia [6] and also Estonia [7, § 36] consider financial services a component of critical infrastructure.

1.1. Motivation and goals

The ease of use is a positive aspect of novel technologies and is the main purpose of contactless payments. However, often there are also additional security and privacy risks involved, especially considering the wireless nature of contactless bank cards. After all, preceding payment technologies, magnetic stripes and “chip-and-pin”, have been success-

fully compromised in the past with card skimmers. How well are the cards with contactless interfaces protected against cloning, fraud or other misuse? One way to solve the posed question would be to systematically check bank cards for irregularities and flaws.

Considering the importance of financial systems, having more research on bank card security open and public, instead in the hands of criminals, will be desirable. Thankfully there has been some research into security of contactless payment systems, however the author could not find examples of systematic fuzz testing conducted on EMV cards or any testing done on Estonian bank cards. This thesis will offer a contribution to address both of those points by devising a systematic approach for testing bank cards and also performing security testing on Estonian bank cards. There are three main goals for this thesis:

1. Test whether Estonian bank cards are vulnerable to known attacks against contactless bank cards.
2. Develop a systematic method for finding irregularities in EMV card behaviour in order to find new vulnerabilities.
3. Test the method on Estonian contactless bank cards

1.2. Scope

While there are numerous contactless payment solutions on the market, this thesis will only look at regular bank cards, excluding any payment tokens, stickers and smartphone based solutions. Research of banking cards is associated with delicate and personal data and therefore deserves special consideration. As there are also potential legal issues concerning the banking networks and hardware, some restrictions will be applied to this thesis:

1. All experiments on the card will be done offline, without interacting with production bank systems (such as payment terminals or ATM-s)

2. Only cards belonging to the author will be examined

The main consequence of this is that certain types of attacks, such as relay attacks on the cards and testing Point of sale (PoS) terminals will not be conducted for this thesis.

1.3. Outline

This thesis is divided between four chapters: Chapter 2 will give background information on the relevant technical standards, Chapter 3 will look at other research done with contactless EMV cards, Chapter 4 will line out the methodology for this thesis and finally Chapter 5 will cover the results and analysis of the thesis.

2. Background

This chapter offers insight into the standards, technical implementation details and a brief overview of the availability of contactless payment solutions in Estonia. This information will help in understanding the methodology and results of this thesis. EMVCo maintains two standards — EMV Contact and EMV Contactless, for chip and swipe bank cards respectively. The latter is based on the former, extending and updating the chip payment standard. It is important to note that EMV Contactless is not designed to replace its predecessor, but to offer an alternative way of performing transactions.

2.1. EMV

An important part of any discussion related to banking cards is the EMV consortium (EMVCo), consisting of world leading payment network providers [8], such as MasterCard and Visa and others. The members of this organization are processing the majority (63.7%) of global transactions [9] and are in charge of many of the standards used in the industry — most importantly those related to contactless card payments. In addition to bank card specifications, the group maintains several standards related to electronic payments — QR codes, mobile payments, 3-D Secure mechanism *et cetera*.

Historically, EMVCo was created in mid 1990s to combat the increasing problem of fraud with stolen bank cards. Previously, the transactions were conducted using data on magnetic stripes or mechanical imprints of the cards, along with customer's signature. The signature could be faked, or with some effort erased and replaced. In more recent history, "skimmer" devices have been used to clone the magnetic stripe data. Europay, MasterCard and Visa created a standard which would transform bank cards into smart cards that have chips in addition to magnetic stripe for backwards compatibility. The chip cards need to be physically inserted to a reader device in order to complete the transaction and allow for card-holder verification via card-holder PIN entry. The procedure of using chip data to complete the transaction is commonly known as EMV, named after the founding companies. The standard has received numerous updates and is now known as EMV Contact, with the latest version being released in 2011[10].

2.2. ISO 7816

Bank cards are in essence smart cards with specialized payment applications — as such, they follow the international ISO/IEC 7816 standard for smart cards. The standard is extensive and is divided to 15 parts, which cover physical attributes, electrical interfaces, data protocols *et cetera*. The most pertinent section of the standard for this thesis will be ISO 7816-4, which defines the protocol for transmitting data to and from smart cards in Application protocol data units (APDU-s) and defines commonly used inter-industry commands.

Communication between card and host is half-duplex — only one party can transmit at one time. Reader is always the initiating party, the card cannot independently send data to the host. For EMV payment cards, the host is normally a PoS or an Automatic Teller Machine (ATM). The host sends a command APDU, the card replies with a response APDU. The cAPDU structure consists of a 4-byte header and up to 65536 bytes of data, an in-depth description can be found in Table 1. The response APDU (rAPDU) has a similar limit of 65536 bytes in payload size and is accompanied by a 2-byte trailer indicating the outcome of the command. rAPDU structure is detailed in Table 2.

	Size (bytes)	**Description**
CLA	1	Class of instruction, also used to signal the logical channel being used
INS	1	Instruction code
P1	1	First parameter for the command
P2	1	Second parameter for the command
L_c	0,1,3	Length of data field, up to 65,536 bytes
Data	L_c	Command payload
L_e	0,1,3	Length of data expected in response

Table 1. Command APDU

	Size (bytes)	Description
Response data	up to L_e	Response payload
SW1,SW2	2	Response status, for example 0x90, 0x00 stands for successful command execution

Table 2. Response APDU

CLA byte Proper understanding of rAPDU field CLA as it is used in Estonian bank cards requires looking at multiple revisions of the ISO 7816-4 specification, limiting oneself to the latest version would leave some results unclear. In the first revision, created in 1995, meaning of CLA is described in hexadecimal value ranges, shown in Table 3. The lower bits ('X') are used to signal secure messaging (bits 3 and 4) and logical channel use (bits 2 and 1).

Value	Meaning
0X	Structure and coding of command and response according to ISO 7816
10 to 7F	Reserved for future use (RFU)
8X, 9X	Structure of command and response according to ISO 7816. Except for 'X', the coding and meaning of command and response are proprietary
AX	Unless otherwise specified by the application context, structure and coding of command and response according to ISO 7816
B0 to CF	Structure of command and response according to ISO 7816
D0 to FE	Proprietary structure and coding of command and response
FF	Reserved for protocol type selection

Table 3. CLA byte in ISO 7816-4:1995 [11]

In 2005 revision of the standard, coding of CLA byte has been changed to a bit-mask, while the functionality provided is the same. The most significant bit (b8) determines whether the command is inter-industry (b8 = 0) or proprietary (b8 = 1). The rest of the bits determine whether secure messaging or command chaining will be used and which logical channel the command belongs to. The maximum amount of secure channels has been expanded to 19, in case b7 has been set to 1. [12, p. 8].

The ISO 7816-4 standard [12] describes several instructions for basic interaction with smart cards. The most important ones for this thesis include `SELECT (0xA4)` for loading applications, `GET DATA (0xCA)` for reading tags and `READ RECORD (0xB2)` for reading one or more records.

2.3. EMV Contact

The payment flow for EMV can get fairly complex, but can be divided into logical phases. These are not always one-to-one mappable to commands sent to the bank card and not all steps are mandatory, depending on the exact circumstances of the transaction. EMVCo has provided a visual overview of the process (see Figure 1)

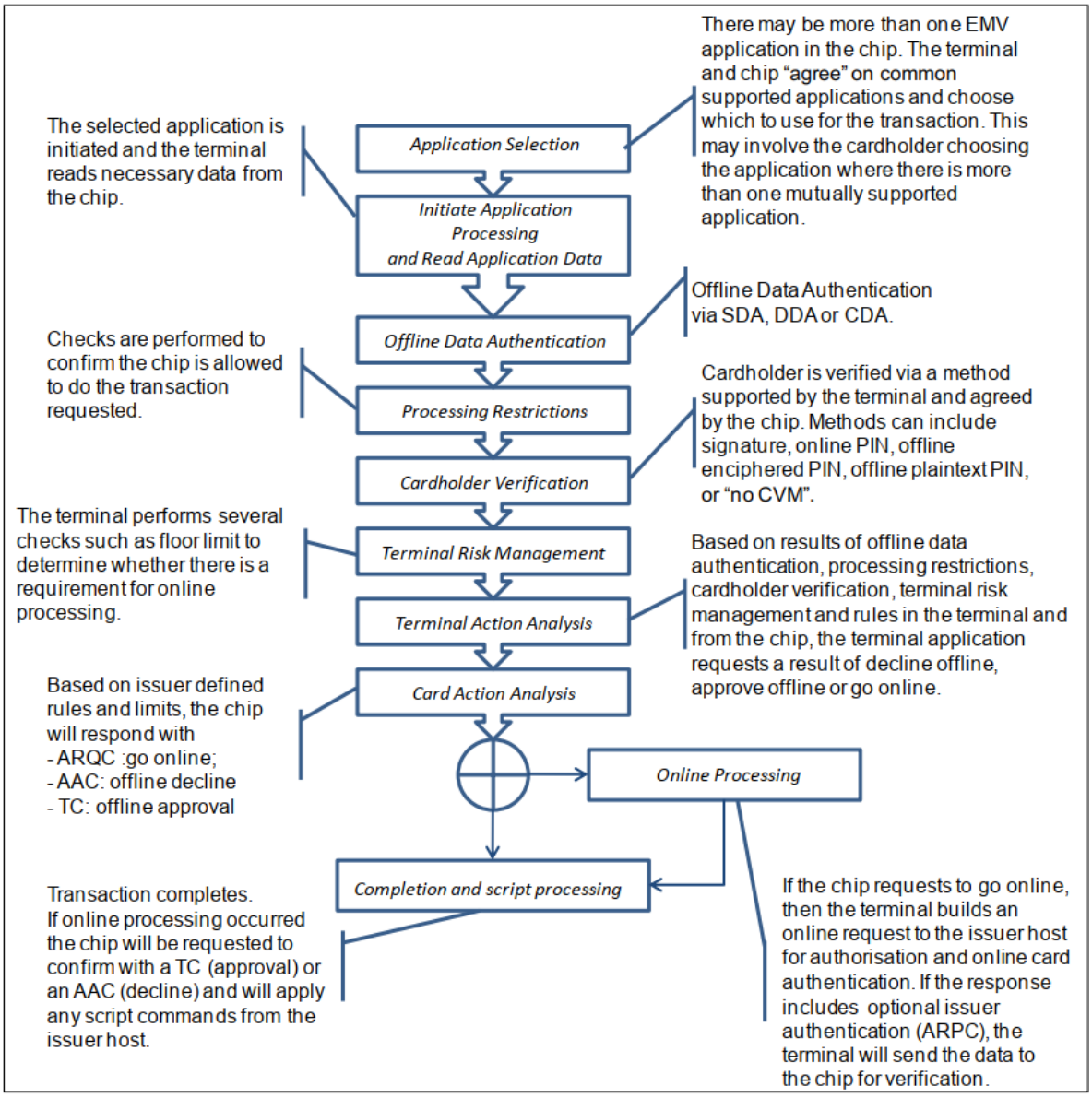

Figure 1. EMV Contact processing flow [13]

For a easier understanding of technical progression in the flow, the author would summarize the EMV Contact to the following steps:

1. **Selecting EMV application** EMV uses ISO 7816 application selection to differentiate between payment products. Each specific product has a specific Application identifier (AID) — a bank card can have multiple products in a single physical card, such as dual use debit and credit cards. The PoS issues `SELECT`command for Proximity Payment System Environment (PPSE) application, which returns a list of payment application AIDs supported by the card. The terminal needs to support one of the returned AID in order to `SELECT` it and procede with the transaction.

2. **Initiate application processing** The terminal sends GPO command to signal the start of transaction flow. The response contains Application interchange profile (AIP) and AFL. The AIP contains information on the card's supported capabilities, such as supported payment flows (Static data authentication (SDA), Dynamic data authentication (DDA), Combined data authentication (CDA)), transaction modes (EMV or Magnetic Stripe), which verification and risk management procedures are to be performed. AFL contains a list of files, which contain information required for completing the transaction.

3. **Read records from the card** Terminal queries the files listed in AFL to obtain data needed for further processing. The information is returned in Tag-Length-Value (TLV) format and includes card expiration date, Primary account number (PAN), Card Risk management DOL (CDOL) lists used for input in generating the transaction cryptogram, risk management profiles, public keys for the ICC and issuer.

4. **Data authentication** If the terminal is working in online only mode, this step is skipped; otherwise one of the data authentication methods will be used to verify the card's authenticity.

5. **Cardholder verification** This step verifies whether the person using the card is its legitimate owner. The method used is called Customer Verification Method (CVM) and there are few possible ways for completing the CVM process: PIN verification (online or offline), signature on paper or no verification at all.

6. **Transaction** Based on risk management rules present in the card and defined by the terminal, restrictions and checks will be applied on the transaction (such as floor

limit, offline payment allowance) and the decision whether the transaction will be online or offline will be made. The card can reject the offline payment and force the transaction to processed by the issuer. The card can also reject the transaction in either case. The process will be negotiated using GENERATE AC commands and their responses.

2.4. EMV Contactless

EMV Contactless specifications consists of four books [14], with the latest update released in May 2018. They are built upon EMV Contact specifications and ISO/IEC 14443 standard to add wireless communication capabilities. The ISO standard defines Type A and Type B devices, both of which use the Radio Frequency Identification (RFID) HF frequency at 13.56 MHz. The types differ in modulation and coding schemes, but follow the same transmission protocol, described in Part 4 (ISO 14443-4). EMV cards can be found in both types. ISO 14443 is a rather commonly used standard — implementations include MIFARE devices, biometric passports and Near Field Communication (NFC). The compatibility with NFC is especially useful as it allows smart-phones and other off-the-shelf NFC readers to communicate with EMV Contactless cards. On top of RFID communication, the higher layers are similar to EMV Contact — smart card commands following the same formats are still being used. A major departure from EMV Contact is addition of different kernels for each EMVCo member:

- Kernel 1 for Visa and JCB
- Kernel 2 for MasterCard
- Kernel 3 for Visa
- Kernel 4 for American Express
- Kernel 5 for JCB
- Kernel 6 for Discover
- Kernel 7 for UnionPay

The payment process is close to EMV Contact, with some notable changes:

1. Contactless payment supports EMV mode and legacy Magnetic Stripe mode for older systems, which cannot use EMV data. This is common in countries such as the US, where chip and pin was never widely adopted and the pre-existing infrastructure only supports transmitting magnetic stripe information.
2. DDA is not used as it requires additional round-trips between the card and reader
3. SDA is supported, but being phased out
4. Only one cryptogram is generated during the transaction

2.5. Contactless bank cards in Estonia

Estonia has been relatively late in offering contactless cards, as the local banks started to issue cards in 2016. The estimations for contactless capable device market share according to Bank of Estonia are 17% for bank cards and 85% for PoS terminals by end of 2017 [2, p. 26], nearly all of the issued cards use MasterCard payment scheme. A brief summary of contactless products follows:

- Pocopay was the first Estonian financial institution to offer contactless payment cards in February of 2016 [15].
- LHV was one of the first banks to introduce a contactless product in August 2016 [16]. At the time of writing, all of their card products are MasterCard brands with contactless support. LHV has an unique feature in allowing cash withdrawals using the contactless interface in their ATM-s.
- Swedbank trialled ISIC student cards in August 2016 [17] and expanded to debit and credit cards in October of the same year. As of writing, only few non-contactless offerings remain — two debit card products for elderly and children. Since the inception of this thesis, Swedbank has also added a contactless Visa credit card to their product selection and has released a testing version of their smart-phone

application for contactless payments [18]. The trend for Swedbank is clearly towards phasing out non-contactless capable card products.

- SEB is the last large bank to introduce a contactless capable card product. Contactless MasterCard debit card was launched in June 2017 [19], but is marketed as an alternative to regular debit cards — the mainstream offer is a regular MasterCard product. SEB also offers contactless capable ISIC and ITIC cards [20].

- Coop bank (merger, formerly Inbank and Krediidipank) is only offering debit cards with contactless payment [21], as did its predecessor Krediidipank.

- Luminor (merger or former Nordea and DNB banks) did not issue new contactless cards after the merger, but previously released Nordea contactless cards stayed operational [22]. In 2018, Luminor has restarted issuing MasterCard debit cards with contactless payment [23].

- Estonian telecommunications company Telia offers a mobile contactless payment application called mTasku, which stores bank card information and can be used to perform payments. In practice, this requires specific support from acquiring merchant and is therefore not as flexible as real contactless payments [24]. As this thesis focuses on contactless bank cards, not phone based solutions, the latter will not be looked into.

2.6. Fuzz testing

Fuzz testing or *fuzzing* is a technique for automated software testing using (semi-)random inputs. It has been very successfully used to find high impact bugs in open source libraries by Google's honggfuzz [25] and afl [26]. Fuzzing works by identifying input points and generating fuzzing data to provoke unintended or buggy behaviour. Fuzzers can be categorized by the way input data is generated — the most common options are mutation fuzzing and randomized fuzzing. Former works by starting with a valid input and performing many test iterations with small alterations in the inputs, latter by generating random input data conforming to the defined restrictions. This thesis will use both approaches in appropriate situations.

3. Related work

EMV Contactless is comparatively new payment technology, but there have been studies on the security of contactless payment process. Documents can be found for portions of the contactless card puzzle — security requirements for JCOP 3 [27], an operating system for smart cards, for example. Common criteria requirements describe the desired functionality and capabilities for the smart card and the operating system, but do not provide a framework for testing whether this has been achieved.

Bulk of the academic material on EMV has been created in and concentrates on Europe, where EMV and contactless payments have been been available for longest. Nevertheless, to the author's knowledge, no studies on contactless bank cards issued in Estonia have been conducted before. An overview of research into contactless card security was conducted, some of the vulnerabilities discovered are listed below.

Harvesting details from contactless cards On-line purchase is one of the Card Not Present (CNP) transaction flows supported by most credit cards. To perform an on-line purchase, usually following pieces of data are required: card number, card expiry date, cardholder name and a verification code called CVV2 or CVC2. This code is usually printed on the back side of the card. Researchers at Newcastle University [28] described a method of collecting card details (cardholder name, card number, expiry date) using the RFID interface. This data being accessible is part of the EMV specification and not a flaw in and of itself, however the paper points out that to only the verification code is missing to have all the required pieces for a complete CNP transaction. In a hypothetical scenario where an attacker has compromised a legitimate PoS terminal or ATM with two additional devices — a contactless reader and tiny camera, all the required details could be gathered with relative ease. Another possibility would be to collect card details without the verification code and attempt to either brute force the code in many on-line retailers or try to find some, which do not validate the verification number. In 2012, popular on-line store Amazon was used for this purpose [29].

This attack is not very reliable, as many conditions need to be filled for a successful fraud

to occur: 1) Use of on-line merchant which does not validate verification codes; 2) 3-D Secure technology is not enabled for the merchant and the card; 3) Brute force verification code scanning is not detected and mitigated by issuing bank.

A similar study was made by a security researcher Renaud Lifschitz in 2012. He points out in addition to generic information about the card, magnetic st ripe data and transaction history is also readable using the contactless interface. Easily accessible transaction history may cause privacy concerns — the log includes timestamp and amount of money involved in the transaction. If the card exposes both tracks of magnetic stripe data, it could be used to construct a magnet-stripe clone. [30]

Offline PIN verification In 2013, researchers at Newcastle University released a paper detailing a more intrusive vulnerability [31]. Their discovery concerns Visa cards, some of which expose plain-text PIN verification command in contactless mode, which is not part of the EMV specifications. In the attack scenario, a hidden harvester device is placed near a RFID door access reader. When people entering the building use their cards to open the doors, they often place whole wallet on the reader, which allows other planted device to try and look for EMV cards with the vulnerability. The regular amount of PIN retries in bank cards is 3, allowing two guesses at the PIN for each use of door reader. As the PIN retry counter is reset on each successful transaction, over time this method could yield numerous correct PIN results. Alternatively, this vulnerability could be used to perform DoS attack by exhausting the PIN retries and blocking the card from use.

Pre-play & downgrade attack Another important method of attack was discovered by Michael Roland and Josef Langer of University of Upper Austria [32]. The attack is based on the legacy Mag-Stripe mode in contactless payments. Due to limited space in magnetic stripe tracks, the Unpredictable Number (UN) and Application Transaction Counter (ATC) have 7 bytes of storage to share between them. This space is split for each issuing bank, however using 3 bytes for UN and 4 bytes for ATC is the most common solution. As the UN is stored in binary coded decimal, 3 bytes of storage comes out to $10^3 = 1000$ possible random values. The attack works by determining the length of UN field and calculating the required amount of CVC3 values. Since ATC field is incremented for each calculation,

all three values (UN, ATC, CVC3) need to be retained. After collecting the values, they can be stored on a cloned device and used with a legitimate PoS terminal. Any UN generated by the terminal will have a match in stored values at least once. The possibility of getting more than one matching UN, ATC combinations from legitimate terminals drops off rapidly. The attack will likely not work if the original card is used before the cloned version, as the incremented ATC has been used in a cryptogram sent to the issuing bank and lower ATC-s will not be accepted.

The MasterCard rules for payment terminals [33, p. 193] require EMV mode, but optionally allow contactless magnetic stripe mode. For Europe region, the rule is more specific — magnetic stripe mode will be allowed until 2020 [33, p. 205]. Accompanying document regarding payment cards [34, p. 17] explains how to mode should be selected — EMV mode is to be preferred, with fallback to magnetic stripe. The described attack will work only work on PoS terminals that support Magnetic Stripe mode transactions, the cloned card would not have required information to perform an EMV transaction. There are very few terminals remaining that only work with Magnetic Stripe mode, however terminals that support both modes could be more common. This is where the downgrade aspect of the attack gets utilized — specification for Kernel 2 does not have a method to directly indicate Magnetic-Stripe mode, however there is a flag for EMV mode support — Bit 8 of second byte in AIP. Should the cloned card set this to be 0, the terminal will attempt to use the Magnet-Stripe payment flow, if supported and cancel the transaction otherwise.

Foreign currency flaw In 2014, Martin Emms *et al.* of Newcastle University published a paper [35], which described a flaw in Visa cards issued in the UK. Contactless payments are protected by a transaction limit that does not allow payments over a certain amount (20£ in the paper) to be finished without performing a cardholder verification, for example with a PIN entry. There is also a offline payment limit, beyond which an on-line issuer authentication needs to be performed. The paper shows how both of these limits can be bypassed by initiating a transaction in foreign currency.

The problem manifests in cards using Kernel 3, which is used for offline mode payments only. In this scenario, authenticated cryptograms for arbitrary sized transactions can be obtained from a victim's bank card without any on-line confirmation by the issuing bank.

The AC does not contain information about the terminal or the merchant, these can be added later as there is no validation for terminal. A scenario is presented, where a rogue merchant is used to convert the transaction data into money, potentially defrauding very large sums.

Relay attack Relay attacks against ISO 1443 devices have been researched on many occasions [36], [37]. The attack utilizes two devices connected with an independent connection, such as Bluetooth, WLAN or even regular Internet. One of the devices functions as a mole, interfacing with the victim's bank card; the second device is a relay, used to communicate with a PoS terminal. In the simplest relay attack setup, the commands from relay are forwarded to the mole, with responses travelling the opposite direction. The relay step adds significant delays to the round trip time between the PoS and contactless card, however it extends the functional range of contactless payments to kilometres, thus breaking the security assurance of short range.

Jordi van der Breekel examined relay attack scenarios on Dutch contactless cards in his thesis [38]. He produced an optimized setup, where both devices used for relay are enhanced with logic to minimize the amount of data required to be relayed over the secondary. This configuration could add as little as 200 ms to the regular transaction, making the attack difficult to detect and mitigate.

Return of Coppersmith Attack (ROCA) In late 2017 a serious vulnerability in RSA keys generated by Infineon chips was found and published [39], dubbed Return of Coppersmith Attack (ROCA). EMV cards also have their Public Key Infrastructure (PKI) set up in RSA format keys and although the specifications dictate that keys should be generated externally, there is always a possibility of corners being cut and weak keys finding their way into bank cards.

Crucial functions of the EMV card such as PIN verification, transaction cryptogram generation and card authentication rely on cryptographic keys. The card contains CA (issuer) public key, Intergrated Circuit Card (ICC) key pair and optionally a second ICC key pair for PIN verification. The primary ICC private key is used to create responses to GENERATE AC

command and is one of the most important pieces of the EMV security puzzle. Any method which allows the compromise of ICC private key confidentiality could lead to functional clones of the bank card in question.

Fuzz testing Fuzz testing or *fuzzing* is a technique for finding defects in software by sending random data to a program's input. There are examples of fuzzing used to uncover bugs in software, such as Google's project Zero [40]. In field of smart cards, information on fuzzing is scarce — the author has not found any examples of academic writing regarding fuzzing on contactless cards. What little material there is can be found in security researcher's blogs and conference presentations.

Peter Fillmore presented at Black Hat 2015 conference [41], giving an in-depth look into contactless EMV security. An open source fuzzing tool is also mentioned, however it has since been removed from the presenter's Github profile [1].

SANS Institute blog hosts a post by Miika Turkia discussing a similar topic — penetration testing payment terminals [42]. He surmises that fuzzing payment terminals (and cards) is difficult due to lack of feedback mechanism on these devices — "The trouble with pen testing production terminals is that one has no information about the internals or what actually occurs inside the payment terminal on crashes. The only thing I know is that the terminal either reboots or goes to an otherwise inoperable state.". He emphasizes the difficulty in creating tools to help with performing automated testing, which might explain the lack of publicly available fuzzers.

Another blog post can be found covering fuzz testing on EMV Contact cards [43], however the linked paper has been removed from the site.

[1] https://github.com/peterfillmore/EMVFuzzer

4. Methodology

As was outlined in section 1.2, only bank cards belonging to the author will be used as test subjects. Cards used for testing were obtained by opening personal bank accounts in Estonian banks, which offer contactless cards and ordering debit cards from each one. At the time of writing the thesis, all but one contactless card products available in Estonia are using MasterCard payment network. The one exception, Visa Platinum by Swedbank, was released during the research period and was cost prohibitive to obtain for the author and is therefore excluded from the test group. The cards used in the test are contactless debit cards from Estonian banks: Coop, LHV, Luminor, Pocopay, SEB, Swedbank. Going forward, the information on specific cards will be anonymised using alphabetical sequence — "Card A" through "Card F". Any vulnerabilities which have been discovered during research have been disclosed to affected banks before the publishing of this thesis.

There will be two main research methods in this thesis:

1. Look at previously discovered attack methods on EMV Contactless cards and test their viability on real bank cards. This will be covered in section 4.2.
2. Find irregularities in contactless EMV bank cards:
   (a) gather information on test cards from readable records
   (b) map the available command space and select candidate instructions for further testing
   (c) determine whether chosen instructions cause irregular behaviour or side-effects when subjected to invalid, unexpected, erroneous or out of bounds input data (fuzz testing) and if it may have an impact on the security of the card

4.1. Equipment and tools used, issues arising throughout the research

In terms of hardware, a rooted Android smartphone (Oneplus One) is used for information gathering and vulnerability testing, as there are many pre-existing NFC and EMV tools

which can be used. The phone additionally serves as a backup for primary research hardware and as a verification tool for real-life experiments. For primary hardware, the author's initial choice was Proxmark 3, which is known for its capabilities with NFC tags. Software and hardware for Proxmark is open source — anyone could use the schematics to build their own device. For ease of use, pre-assembled and tested hardware kits are also available for purchase. In order to start the initial testing of the bank cards, a previously assembled device was loaned from the university. Proxmark has had support for ISO 14443A communications for a while, whereas EMV specific commands were added only in late 2017 [44]. When experimenting with the Proxmark console and API, it became apparent that automation of any kind will be a tedious proposition. While Proxmark can be scripted using Lua language, the module for ISO 14443 and EMV exposed very small API surface to work with and no opportunities to programmatically parse the responses from the card. Unfortunately the only way to use the hardware is through the original software, no third party tools can be utilized.

Additionally, more problems rapidly appeared when attempting to complete a payment flow — when issuing GPO, the card sent erroneous response of `fa 00 01` (see Appendix A) and the connection became unresponsive. This does not match any known error code, indicating a problem with the hardware used. To confirm the issue was restricted to Proxmark device itself, identical commands were sent to the bank card using a NFC-enabled Android device (see Figure 2), which worked properly. Next guess was that the specific Proxmark device in use is faulty — in order to confirm this assumption the author loaned another Proxmark device to test with. The second device unfortunately produced the same results. Due to these major hurdles, further attempts of using Proxmark were abandoned.

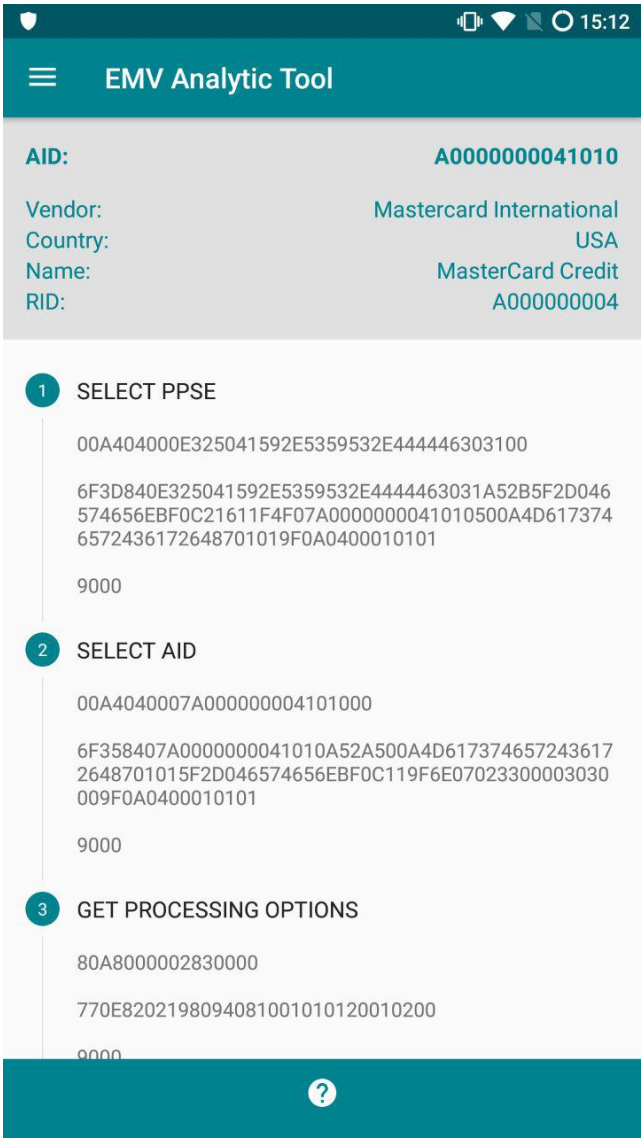

Figure 2. GPO example from Android phone

The author also attempted to use Proxmark device to perform eavesdropping on RFID traffic between an Android device and a contactless card using the `hf 14a snoop` command. The eavesdropping mode was often unstable and usually captured only traffic from the device physically closest to the reader, with only fragments of the second portion. While observing the captured data, numerous Cyclic Redundancy Check (CRC) errors were present, which makes the accuracy of the results doubtful.

To proceed with the experiments, a Personal Computer/Smart Card (PC/SC) compatible NFC reader ACR1252U was purchased. The PC/SC interface is a common API for accessing smart cards. Many programming languages can use this interface and helpful libraries and software are available in open source. One such tool is Cardpeek [45], a scriptable smart card reader software with many packaged scripts to read from e-Passports, Traffic cards, EMV *et cetera*. The built-in EMV script was used to profile the card for common EMV information, and to perform initial tests for some of the known vulnerabilities. The programmable API for Cardpeek is written in Lua language, enhanced with specialized libraries for smart card communication. Although useful for writing small scripts, the limitations of Lua language make Cardpeek unsuitable as a fuzzing tool.

MIT Lincoln Laboratory has an open-source library for smart card scripting called LL-Smartcard [46], also using PC/SC. The library is using Python language, which is more flexible and familiar to the author, for this reason LL-Smartcard will be used as basis for writing fuzzing scripts.

To summarize, the list of hardware and software used in this thesis is:

1. ACS ACR1252U USB NFC reader
2. Oneplus One smartphone
3. Cardpeek (software)
4. pcsc-tools (software)
5. fuzzing scripts based on LL-Smartcard (software)

Additionally, `scriptor` command from `pcsc-tools` package was used to test individual commands.

4.2. Applying previously discovered attack methods on test cards

Several attack methods have been described in Chapter 3, however not all of them are suitable to be tested on the test cards group. Namely — two vulnerabilities related to Visa cards **offline PIN verification** and **foreign currency flaw** have to be ruled out since none of the test cards are using Visa payment network.

4.2.1. Key infrastructure

Return of Coppersmith Attack (ROCA) The test will consist of reading ICC public key through READ RECORD command, which is exposed after performing the `GET PROCESSING OPTIONS`. The test can be performed using an open source Android application [47].

The android tool only checks ICC public key, additional verification will be done on a PC. Firstly, a tool [48] will be used to extract all RSA keys from the card (including issuer key) in PEM format. Secondly, original ROCA researchers produced an open source tool `roca-detect` [49] for checking vulnerable certificates, which will be used to verify the results from the android application.

4.2.2. Magnetic stripe data — pre-play attack

The attack described by Roland and Langer is not overly complex and can be tested in a script format fairly easily. There are also open-source Android applications available [50], [51], which implement both phases of the attack — gathering of CVC3 values and doing card emulation.

First of these, EMVEmulator has several problems which prevents it from being useful: it does not parse Track 2 bitmaps and blindly assumes the UN length of 3, which can be incorrect. This can be wasteful if real UN length is different — for larger UN length, this attack becomes impractical as not enough combinations are calculated; for smaller, computing 1000 cryptograms is inefficient. Secondly, the application works only when in

foreground, it cannot persist card data across many runs.

The second application called MasterTap looks much more promising. Scanning the card stores it in long term storage, which can be protected by a PIN, and export to file functionality is also present. Correct UN length is also detected, as opposed to EMVEmulator.

4.2.3. Magnetic stripe data — harvesting

Harvesting magnetic stripe data is probably not effective, as the problem of harvesting magnetic stripe data has persisted since EMV Contact was introduced. In addition to increased awareness of the problem, many online merchants are using additional checks, such as 3-D Secure and CVV verification to avoid fraudulent transactions.

The NFC interface on bank cards exposes Track 1 data (tag 56) and Track 2 equivalent data (tag 57), without the CVV values. Ignoring the CVV difference, according to previous research this does not match the magnetic stripe data on the card exactly [52], [53]. Due to lack of magnetic stripe encoder/decoder, this claim is left unverified. To check the viability of this method, a CNP payment with the gathered data will be attempted.

4.2.4. Relay attack

Relay attacks require a working PoS terminal to complete, along with two proxy devices and the card to be attacked. This would fall outside of the scope defined in the section 1.2, and will not be performed as part of this thesis. In theoretical terms, new versions of MasterCard standard describes resistance protocol [54, p. 97], which establishes timeout thresholds through for processing. This allows the detection of additional processing nodes in chain, but the feature only works when both the card and PoS indicate support.

4.3. Systematic method for finding irregularities in EMV cards

The EMV payment process can be broken down to several commands, which are issued to the card in order to complete the payment flow. Some of these commands alter the state of the card so that the set of usable commands changes. Some of these state transitions uncover new commands, while others make little changes. For purpose of this thesis, the process is divided in three stages:

1. Card entered into reader field, before SELECT command has been issued to the card. No applications have been loaded in this stage, issues with base operating system for the bank card could potentially be exploited at this point.
2. Payment application selected (MasterCard or equivalent kernel loaded). In this environment the potential impact will be greater as data necessary for transactions will be accessible. Additionally, this stage is likely to be more customised by the issuing bank to add or enable special features in the card product. This also increases the chance of bugs being introduced.
3. GPO issued, card ready for processing transaction. The security implications in this stage are similar to previous one.

Each of these states will be used as a starting point for mapping the available command space and finding candidate instructions for further testing. The author did not find a good pre-existing tool to accomplish the various steps to be discussed in this section and created a script in python to accomplish many of the tasks listed. The source code for this script will be available at GitHub [55], it is based on the open-source LL-SmartCard library and is written in Python language. There are command line arguments for selecting the reader, debug mode (outputs all APDUs exchanged with the card), choosing a logical channel and issuing preset commands before other operations (such as selecting MasterCard application and/or issuing GPO).

4.3.1. Information gathering through known methods

In addition to looking at the commands available, the thesis will look at other information readable from the card, using two standard commands — `GET DATA (0x80 0xCA)` and `READ RECORD (0x00 0xB2)`. Both commands return results in TLV encoding, consisting of tag (description of the record), length of the payload and actual data. Tags can be nested inside the value portion, used for example in CDOL.

GET DATA `GET DATA` is used to retrieve the value for specific tags (submitted as P1 and P2), meaning up to 2^{16} tags can be directly queried by the command. EMV specification defines few tags, which are commonly used as part of payment flows — such as PIN retry counter (9F17), ATC (9F36) and Log format description (9F4F). The information gathering step will iterate over all possible parameters to determine whether additional tags have responses. The author's script can be used to perform this step in all stages:
stage 1: `$ bin/fuzz -r 0 -target 0x80 0xca`
stage 2: `$ bin/fuzz -r 0 -mc -target 0x80 0xca`
stage 3: `$ bin/fuzz -r 0 -mc -gpo -target 0x80 0xca`

READ RECORD `READ RECORD` command is used to read files/records from the card. These files are organized into 31 Shoft File Identifiers (SFIs), ranging from 1 to 31, each of which contain up to 256 files. GPO response contains AFL tag, which contains list of files required for the transaction processing. Normally, this list contains 3-4 files. The total number of files is small enough to be exhaustively checked — this step will be accomplished using Cardpeek software, which already includes this functionality in its EMV preset. This step is not done in stages, as the AFL information is only retrieved in the final stage.

4.3.2. Command mapping

Command mapping is done in two steps — firstly looking up available instruction classes by issuing APDU with INS, P1, P2 equal to 0x00 and varying the value for CLA from 0x00 up to 0xFF. Filtering out the responses which did not respond with "CLA not supported" response

(SW1-SW2 6E 00) leaves the set of supported command classes. For each supported class, another pass will be conducted where CLA value is fixed and INS will be varied from 0x00 up to 0xFF, P1 and P2 staying as 0x00. Similarly, the results will be filtered out based on "INS not supported" response (SW1-SW2 6D 00).

The command mapping mode is the default when running the author's script:
stage 1: `$ bin/fuzz -r 0`
stage 2: `$ bin/fuzz -r 0 -mc`
stage 3: `$ bin/fuzz -r 0 -mc -gpo`

At this point, list of supported commands is ready, which can be cross-checked with EMV and ISO7816 documentation to try and match with known commands. Some of the detected commands will have no immediate match, which merit additional testing.

4.3.3. Profiling outlier commands

For unknown commands, it may be required to iterate through the remaining header fields to reach a response without errors. In each step, the response code may hold clues towards the valid command signature. The first fields to cover will be parameters P1 and P2 — for many commands, only specific P1 and P2 values will be accepted while others accept all input without discrimination. SW1 value of 6a or 6B signifies invalid parameters, in which case SW2 can be helpful to pin-point the problem. The second code can also be very generic, such as 00, which translates to "no information given, P1 or P2 incorrect". The total number of possible P1-P2 combinations is relatively small, therefore entire parameter space can be tested and valid parameters retained for the next step.

For the commands which accept input of specific length, SW1 of 6C or 67 is returned for commands with wrong length. With 6C, the correct length is provided in SW2; with 67, the valid length needs to be determined via trial and error method.

Using the author's script, these steps can be accomplished as follows: finding correct parameters: `$ bin/fuzz -r 0 -target 0x80 0x0e`

testing payload lengths: `$ bin/fuzz -r 0 -d -target 0x80 0x0e -target-length 2`

4.3.4. Command selection

After the set of working commands is found and the signature for parameters and length is determined, a choice has to be made on which commands will be tested further. Certainly, commands outside of 7816 and EMV specification will be included, since these can yield interesting results. For commands with known functionality, good candidates for testing are those which accept a large payload for input or perform significant computation.

GENERATE AC A fine example of a command matching this description would be `GENERATE AC (0x0 0xAE)`, used to create the payment cryptogram. This command is certain to be present in every EMV card and uses a lengthy CDOL as input data, which is a good target for mutation testing and incorporates cryptographic calculation on the card as part of the signing process. For these reasons, this command will be included as a fuzzing target. There is a drawback looming — each generated cryptogram increments the internal ATC counter, which is capped at a fixed value. Hitting this value will likely render the card unusable for further transactions.

PUT DATA For other commands, those with no known payload structure, the data sent to the card will have to be generated randomly. One such command is `PUT DATA (0x80 0xDA)`. The command accepts any input and supports writing this data to five tags — `9F75` to `9F79`. The amount of data storable varies by card, one source puts the maximum length at 192 bytes [56].

4.3.5. Payload selection for fuzz testing

Fuzz testing or fuzzing is a technique for testing software which employs random, invalid or unexpected data as inputs. Software being fuzzed can crash, leak memory or start behaving in unforeseen way, however testing smart cards is a very opaque process — there is no clear method of reading the internal state of the hardware and determining the results for fuzzing has to be done in an indirect manner. The most obvious feedback mechanism for smart cards is becoming unresponsive or dropping the connection to reader device — this could mean a crash in the operating system or running application. Another method for gaining insight will be to look for irregularities in response codes or time needed to execute the command or determine if any other commands are behaving differently.

Input generation is one of two types:

1. generate completely random data, this will be suitable for unrecognised commands. This can be done using the author's script:
   ```
   $ bin/fuzz -r 0 -mc -gpo -target 0x80 0xda -target-length 32 -p1
   0x9f -p2 0x76
   ```
2. start with a valid input and use mutation to slightly alter it to provoke errors. This will be suitable for use with recognized commands, such as interindustry section of ISO 7816-4 or those belonging to the EMV specification. Mutation fuzzing can be done using the author's script:
   ```
   $ bin/fuzz -r 0 -mc -gpo -target 0x80 0xAE 0x90 0x00 -payload 0x00
   0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x02 0x33
   0x80 0x00 0x00 0x00 0x80 0x09 0x78 0x18 0x03 0x25 0x00 0x00 0x00
   0x00 0x00 0x21 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
   0x1F 0x03 0x02 0x15 0x50 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
   0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
   0x00
   ```

There are open source tools available, such as afl [26] or Sulley [57], to ease the fuzzing process. Unfortunately, both examples listed are suited to work with binaries or networked servers (in case of Sulley) and not very helpful when working with smart cards. MIT

Lincoln laboratory has a tool for fuzzing NFC applications called LL-Fuzzer [58], which is more related to the task at hand, however requires specialized hardware and is limited to fuzzing pure NFC devices.

5. Results and analysis

5.1. Applying previously discovered attack methods on test cards

5.1.1. Key infrastructure

The ICC public key sizes for the cards range from 896 to 1408, former being the most common. Key sizes of 1024 bits or lower have not been recommended for SSL/TLS certificates starting from 2010 [59]. Mozilla, for example, has dropped support for site certificates signed by 1024 bit CA-s since 2015 [60]. Using a a web site, which aggregates sources from multiple academic and private organisations [61], it becomes clear that the recommended key size is significantly higher than the range used in the tested bank cards. While there is no immediate security risk, the key size should be increased in newly issued cards to avoid any potential threats. The key sizes for tested cards are listed in Table 4.

Card	**Key size**
Card A	1152 bits
Card B	896 bits
Card C	1024 bits
Card D	896 bits
Card E	896 bits
Card F	1408 bits

Table 4. ICC key sizes

Return of Coppersmith Attack (ROCA) Using the smart phone application to check ICC keys showed no vulnerable entries. A secondary verification was also performed, to double check the results from the app. The `emvkeytool` tool will use `READ RECORD (00 B2)` command to retrieve the RSA public keys from the card and save them in Privacy Enhnanced Mail (PEM) files. Each card contains issuer public key and ICC public key. `roca-detect` tool was used to check the stored files and none were vulnerable to ROCA.

5.1.2. Magnetic stripe data — harvesting

It is possible to read Track 1 and Track 2 data from the test cards, but in all instances the cardholder name was absent in the response. Alongside other shortcomings mentioned in section 4.2.3, it is definitely not possible to recreate the magnetic stripe from data accessible through NFC interface.

As outlined in section on scope (1.2), no real transactions will be attempted. It is possible to observe the security requirements for completing a CNP transaction in various online vendors. There was a single example, Amazon, which did not require CVV code or cardholder name while adding a payment card to the profile. The user has to provide their full name while completing the account, which will likely be used with the purchase. Fraudulent purchases are often charged back at no cost to the victim, so a company of Amazon's size is using other fraud protection methods to compensate for the lack of CVV verification.

5.1.3. Magnetic stripe data — pre-play attack

Of the tested cards, only one had support for Magnetic-Stripe mode when using contactless interface. This is determined by the presence of a file with magnetic stripe tags present (usually SFI 1 record 1). PayPass rules dictate all issued cards must support both EMV and Magnetic Stripe mode until 2020 [33], which means the other bank cards are in violation of this clause.

When looking at the supported card, the UN bitmap indicates 2 bytes are reserved for UN in UN+ATC section of track 2. Since the UN is encoded using binary coded decimal system, this leaves only 100 combinations of UN-s. Using a script to calculate the UN worked without problems when using a desktop PC with NFC reader. A 3rd party Android application can be used to automate the cloning and re-use process [51]. Cloning of the card using MasterTap application was successful.

Before conducting an experiment with cloned card information in real life situation, the

issuing bank was contacted and informed of the potential problem. During the meeting with the bank, a cloning experiment was performed using a test card and terminal provided by the bank. The devices are supposedly similar to publicly used ones and the result of the experiment would be indicative of real life test. In the experiment, the card cloning worked as with the author's personal card, however payment using the cloned data was not successful. The discussion after indicated that the terminal in use does not support legacy Magnetic-Stripe mode payments, same as most terminals in use in the Baltic region.

To conclude — the card is indeed vulnerable to pre-play cloning, however the downgrade aspect of the attack (forcing the terminal to work in Magnetic-Stripe mode) is not viable in Estonia. No guarantees exist on terminals in other territories, which still may support the legacy protocol. One of the proposed mitigations in the original paper — tracking large ATC jumps — has not been implemented for the Estonian bank.

5.2. Systematic method for finding irregularities in EMV cards

As outlined in section 4.3, test cards were examined in three stages: before selecting the payment application, before issuing GPO and ready for transaction.

5.2.1. Information gathering through known methods

GET DATA The cards responded with a multitude of tags — 107 distinct tags were detected across all cards and all stages. Cross checking with EFTlab's EMV & NFC tag list [56], there are 16 matches with kernel 2 (MasterCard) specific tags, 48 matches with generic EMV tags and 43 tags with no match. The breakdown of valid tag response counts can be found in Table 5.

	stage 1	stage 2	stage 3	Total unique tags
Card A	17	30	30	36
Card B	11	74	37	84
Card C	11	74	74	84
Card D	11	74	74	84
Card E	11	82	38	82
Card F	6	73	73	78

Table 5. Number of valid `GET DATA` responses

The amount of data exposed by `GET DATA` command is predictably low in stage 1 — up to readable 17 tags. These tags contain generic information about the chip, such as Card Production Life Cycle (CPLC) (tag 9F7F), Key information template (tag E0) and other card data (tag 66) [62]. As the payment application has not been loaded at this point, no EMV specific information is present. After performing the `SELECT`, the variety of readable tags is greatly expanded — the number of tags ranges from 30 to 82. In third stage, most cards retain the tag configuration from before GPO was issued, with two exceptions — cards B and E.

The differences in the behaviour for `GET DATA` might be caused by the choice of chip used in the card's manufacture, the operating system and the level of customisation done on the card. When looking at the amounts of tags returned across all stages, some patterns and similarities start to emerge.

Cards B to E respond with identical set of tags in the first stage, suggesting a similar base chip used for these cards. Cards C and D continue to behave similarly across the second and third stages. Card B is similar to C and D in stage two, but has a much smaller data footprint in stage 3. Card E's list of tags grows in stage 2, however in stage 3 follows the example for card B in reducing the amount of readable tags, the only examples to do so. Card A is behaving quite differently from the rest of the test cards by having a compact set of commands in later stages. In contrast to others, ATC, PIN retry counter and other common EMV tags are absent in `GET DATA` responses for card A. PIN retry counter and

log format are instead included in the response message for GPO. Card F likewise diverges by starting with a tiny set of readable tags, which grows in second stage to 73. While this number is similar to the most common in this stage (74), the readable tags for card F are somewhat different from cards B, C and D.

The author surmises that cards A and F are distinct from the rest of the group and are likely using a different chip, different operating system, or are heavily customised. Meanwhile cards B to E are all based on similar hardware and exhibit varying levels of customisation, with cards C and D being the closest in this regard.

READ RECORD The AFL responses for the test cards fall into two groups: card C, which contains four records and all the other cards, which respond with three records. The exact records returned can be found at Table 6. The additional record for card C contains the required tags for magnetic stripe mode payment. By scanning through all possible records, cards A, B, C, D and F also respond with transaction logs in SFI 11. The transaction log stores last 10 transactions made by the card — date and time, amount, currency, CVR, ATC. The transaction log contains entries for both chip and contactless payment interfaces. The interpretation of the log relies on the log format tag (9F4F), readable using `GET DATA` command, for two of the listed cards (B and D) this tag is not present, however every card follows the same log format.

Card A	SFI 2 record 1, SFI 4 records 1-2
Card B	SFI 2 record 1, SFI 4 records 1-2
Card C	SFI 1 record 1, SFI 2 record 1, SFI 4 records 1-2
Card D	SFI 2 record 1, SFI 4 records 1-2
Card E	SFI 2 record 1, SFI 4 records 1-2
Card F	SFI 2 record 1, SFI 4 records 1-2

Table 6. AFL entries

5.2.2. Command mapping

Stage one — before selecting payment application The most interesting results of command mapping were found in the first stage, before EMV application is loaded. Mapped instructions from stage one can be found in Table 7. All tested cards expose `SELECT (0x00 0xA4)` command to start the payment process by selecting the MasterCard application and `GET DATA (0x80 0xCA)` command for reading data from the card. The latter was separately analysed in section 5.2.1. Cards A to E also expose the interindustry version of the same command, `GET DATA (0x00 0xCA)`, which behaves identically. Cards A to E additionally expose command `0x80 0x50`, which will be covered in section 5.2.5. Card A has a similar command with a different **CLA** value of `0x84`, which will be covered in section 5.2.5. Cards A to D expose commands related to managing logical channels (`0x00 0x70` and `0x10 0x70`), covered in section 5.2.3.

One of the tested cards, Card F, has significantly different set of commands responding to the mapping, exposing only the minimum amount necessary to perform the EMV payment transaction. This is in contrast to all others test subjects, which have varying amount of superfluous commands available. Cards B to E expose a set of commands — `0x80 0x0E`, `0x80 0xD8`, `0x80 0xE0`, `0x80 0xE2`, `0x80 0xE4`, `0x80 0xE6`, and `0x80 0xE8`, most of which (except `0x80 0xD8`) are defined in ISO 7816-4 as card lifecycle management commands. Mapping the usable parameters for those commands yielded only Conditions of use not satisfied (`0x69 0x85`) responses, leading the author to believe there exists a distinct configuration mode, entered by following the personalization process for GlobalPlatform smart cards [63, p. 300], which can be initiated with `INITIALIZE UPDATE (0x80 0x50)`. It appears that cards B through E are using similar operating system or card customisation, which leaves many of the configuration commands visible on the card, whereas cards A and F are different in this regard.

	Card A	Card B	Card C	Card D	Card E	Card F
0x0 0x70	present	present	present	present		
0x0 0xA4	present	present	present	present	present	present
0x0 0xCA	present	present	present	present	present	
0x10 0x70	present					
0x80 0x0E		present	present	present	present	
0x80 0x50	present	present	present	present	present	
0x80 0xCA	present	present	present	present	present	present
0x80 0xD8		present	present	present	present	
0x80 0xDA		present	present	present	present	
0x80 0xE0		present	present	present	present	
0x80 0xE2		present	present	present	present	
0x80 0xE4		present	present	present	present	
0x80 0xE6		present	present	present	present	
0x80 0xE8		present	present	present	present	
0x84 0x50	present					

Table 7. Command mapping in stage one

Stage two — after loading MasterCard application The differences in commandsets are greatly reduced after the MasterCard kernel has been loaded. The results can be seen in Table 8. As in previous stage, Card F only exposes the commands necessary to proceed in EMV payment flow and nothing else. Cards A to E preserve some of the commands from the first stage — `GET DATA (0x80 0xCA)`, `PUT DATA (0x80 0xDA)` and commands related to logical channels. An unknown command `0x80 0xD2` appears in this stage for cards A to E, however regardless of parameters used, it responds with `0x6A 0x86`, incorrect P1 or P2 parameter.

	Card A	Card B	Card C	Card D	Card E	Card F
0x0 0x70	present	present	present	present		
0x0 0x84	present	present	present	present	present	present
0x0 0xA4	present	present	present	present	present	present
0x0 0xB2	present	present	present	present	present	present
0x10 0x70	present					
0x80 0xA8	present	present	present	present	present	
0x80 0xCA	present	present	present	present	present	
0x80 0xD2	present	present	present	present	present	
0x80 0xDA	present	present	present	present	present	

Table 8. Command mapping in stage two

Stage three — after issuing GPO In the last stage, when `GET PROCESSING OPTIONS` has been issued, the card is ready to process transactions. The logical channel management commands remain unchanged from stage two. The unknown command, `0x00 0x2D` also stays enabled, albeit inert. The rest of the new command in this stage are as follows:

- Cards A to E expose `COMPUTE CRYPTOGRAPHIC CHECKSUM (0x80 0x2A)` command, used by the Magnetic Stripe mode to calculate the CVC3. Only one of the tested cards, card C, actually supports the Magnetic Stripe mode, for others this command should not be there. Attempts to use the command on other cards are not successful — using valid input only results in `0x69 0x85` as responses.
- Cards B-E expose `INTERNAL AUTHENTICATE (0x0 0x88)` command, which is part of the DDA authentication schema in EMV Contact specification [64, p. 65]. It is not used in EMV Contactless transactions, however INTERNAL AUTHENTICATE is also described as inter-industry command in ISO 7816-4 [12, p. 50], where it is used to authenticate the card to the reader. On the tested bank cards, using the command with data length other than 4 gave the response `0x67 0x00` (wrong data length). Using any 4 byte payloads yields `0x6A 0x81` (function not supported) responses for all tested cards.

	Card A	Card B	Card C	Card D	Card E	Card F
0x0 0x70		present	present	present		
0x0 0x84	present	present	present	present	present	present
0x0 0x88		present	present	present	present	
0x0 0xA4	present	present	present	present	present	present
0x0 0xB2	present	present	present	present	present	present
0x10 0x70	present					
0x80 0x2A	present	present	present	present	present	
0x80 0xAE	present	present	present	present	present	
0x80 0xA8	present	present	present	present	present	
0x80 0xCA	present	present	present	present	present	
0x80 0xD2	present	present	present	present	present	
0x80 0xDA	present	present	present	present	present	
0x80 0xEA					present	

Table 9. Command mapping in stage three

Looking at discovered commands across all stages, instructions that are outside of EMV flow are summed up in Table 10. Having additional instructions present on the card in and of itself is not a security problem, however it does represent a potentially increased attack surface.

5.2.3. Logical channels

Logical channels are supported by four out of six cards tested — A, B, C and E. All of them have the `MANAGE CHANNEL (0x0 0x70)` command, which can be used to open or close a logical channel. Each card supports channels 0 to 3. As descirbed in section 2.2, the least significant bits of CLA are used to indicate the logical channel for the command. This was confirmed by testing — for example opening channel 2, and performing command mapping

CLA	INS	Command name	Description
0x00	0x70	MANAGE CHANNEL	Open or close logical channel
0x10	0x70	MANAGE CHANNEL	Open or close logical channel (with secure messaging CLA)
0x80	0x50	INITIALIZE UPDATE	Start card personalization process [63, p. 300]
0x84	0x50	INITIALIZE UPDATE	Same as above, more details in section 5.2.5
0x00	0x0E	ERASE BINARY	Delete a single file
0x00	0xD2	WRITE RECORD	Write to / update record
0x00	0xD8	unknown	
0x00	0xE0	CREATE FILE	Create new file
0x00	0xE2	APPEND RECORD	Add a record to end of file
0x00	0xE4	DELETE FILE	Delete a single file
0x00	0xE6	TERMINATE DF	Terminate ("Termination state — the object is logically reduced with restricted capabilities or functionality but selectable" [65, p. 3]) file
0x00	0xE8	TERMINATE EF	Terminate file

Table 10. ISO 7816-4 commands

showed additional commands, with the adjusted CLA byte (0x2 and 0x82). Opening a logical channel did not change the commandset available, however attempting to use any of the commands (for example selecting the MasterCard payment application) in a logical channel was not successful — error returned was `CLA not supported (0x6E 0x0)`.

Card A exposed additional command related to logical channels — `0x10 0x70`. According to 2005 revision of ISO 7816-4, this indicates secure messaging. The command had slightly different behaviour compared to `0x0 0x70` — where the latter exposed new commandset based on the channel opened for all pre-existing command classes, the secure messaging variant did not open logical command versions of CLA `0x84`.

5.2.4. Interesting exposed commands

ERASE BINARY One of the commands found in cards A to E is `00 0E`. It is not part of any EMV flow, however ISO 7816-4 describes the command as ERASE BINARY. It is uncertain, if the ISO description matches the behaviour on the sample cards, but assuming it does, unfettered access to erase functionality on the card would allow for an easy option of performing DoS attacks anonymously. The command on cards matches data length described in specifications, testing through all possible P1, P2 combinations along with randomized data value only yields a single type of response — preconditions not filled (`0x69 0x85`). This could mean that there is a possibility to issue certain commands to achieve such a state in which ERASE BINARY could work.

5.2.5. Randomized fuzzing

INITIALIZE UPDATE When the command `0x80 0x50` was found on test cards, it was not possible to find references to such instruction in the EMV specification or the ISO standard. Using trial-and-error testing, the command length was determined to be 8. The card accepts any 8 bytes of data and returns a success code and a deterministic data payload — last 8 of 28 bytes change based on given input. After coming across a document describing the personalization process for GlobalPlatform smart cards, the command signature matched `INITIALIZE UPDATE` command. The personalization process is performed after the card has been manufactured, but before it is loaded with specific information regarding the final card-holder. This information includes keys for PKI infrastructure, account details, PINs, *et cetera*.

As the name suggests, this command should merely initiate the personalization process, there are follow up commands for performing authentication (EXTERNAL AUTHENTICATE) and writing data to the card. The command for authenticating the personalizing device requires the keys embedded during the manufacture of the cards. The key size of 16 bytes (128 bits) makes it infeasible to brute-force the correct payload for continuing with the personalization process. It is nevertheless significant, that the command for performing card personalization is not disabled in production bank cards. Should the secret key

be obtained through leak or other means, it may be possible to enter the personalisation mode and read or alter the personalised data stored on the card.

INITIALIZE UPDATE on Card A On one of the cards, Card A, an interesting variation of the previously discussed command was found. `0x84 0x50` is similar to INITIALIZE UPDATE, but uses different instruction class. According to CLA bitmap covered in section 2.2, CLA of 0x84 means: command chaining enabled — not the last command; logical channel 8. However this card only supports logical channels in range 0-3, suggesting misuse of CLA bitmaps. If b7 is set to 1, but the table for b7=0 is used (as indicated by logical channels 0-3), the CLA will decode to: command chaining, not the last command; proprietary secure messaging format. It is unclear, which interpretation is intended in this case.

Attempting to map the parameters revealed that the process time is unusually high — around 3 seconds for each command. Secondly, after attempting the parameter mapping, something altered the internal state of the card — namely error responses to unsupported commands in valid CLA namespaces changed from `INS not supported (0x6D 0x00)` to `CLA not supported (0x6E 0x00)`. Issuing further `0x84 0x50` commands does not seem to alter this behaviour. In this altered state, the card is still functional in both contact and contactless mode.

PUT DATA For all cards that support PUT DATA command (A to E), there are five writeable tags — `9F 75` up to `9F 79`. The command accepts input with length of up to 32 bytes — lower than the maximum value found from literature. Nevertheless, only one of the tested cards (Card A) rejects actual longer input with an error — others return success with up to 500 bytes of extra input.

For card A, 15000 iterations of randomized 32 bytes were used for testing with no anomalous results. Other cards were also tested with 15000 iterations of randomized input, length field was set to 32, however actual input varied between 300 and 500 bytes. Cards B and D started producing "invalid precondition" (`0x69 0x85`) responses after 1500 iterations and continued to do so until the end of the run. Cards C and E did not produce

anomalous results.

After performing the write test, a read back from the cards was attempted. The data written to the tag specified in `PUT DATA` can be read back with no problems. Reading from other accessible tags returns empty values for most cards, as expected. For card D other user writeable tags contained 3 bytes of data, seemingly the default value. This shows that while the cards may accept larger than specified inputs for PUT DATA command, it is processed securely and no buffer overflows were triggered.

5.2.6. Mutation fuzzing

GENERATE AC The main payload for `GENERATE AC`, the CDOL, is generated according to the template given by the card in the initialization phase, in one of the SFI records referenced in AFL. For all cards tested, the same CDOL format was returned, therefore a common initial fuzzing target was employed for each card. The exact APDU is detailed in Appendix B. There were five runs of testing completed on each, unless noted otherwise. Since generating cryptograms increments transaction counter, each run of fuzzing was limited to 500 iterations of mutation, each iteration mutated 3 to 10 bytes by adjusting the current decimal value by random amount (-64 to 64). The cards do not allow consecutive cryptograms to be created and require a soft "reset" of state between each `GENERATE AC`. This can be achieved by re-issuing `GET PROCESSING OPTIONS` command. For most test cards, the initial invocation for the command would fail with "invalid precondition" (`0x69 0x85`) response, however issuing second GPO would properly reset the card state and enable generating another cryptogram.

It was common for the test input for the command to mutate into invalid state, producing "invalid precondition" responses. In rarer instances, the input would change into something that would turn the card unresponsive or drop the connection. The card behaves like a black box and it is not possible to assess the internal state when this happens. After restarting the process by physically removing and replacing the card and attempting the last input, the cards behaved normally and did not crash. This leads the author to believe that the crash condition is caused by many factors, not a single command invocation. This problem was particularly prevalent with card E — runs for them were extremely

short, often only less than 10 commands were successfully tested before the connection was dropped. To achieve comparable results for card E, additional fuzzing sessions were conducted.

Aside from response code, timing information was also observed to determine any inputs which cause abnormally long processing times. The fuzzing timing results can be found in Table 11. As can be seen from the table, all the averages for all tested cards were around 200 ms per `GENERATE AC` command. The timing was fairly consistent with low standard deviation from the average. Card D stood out from the others with high standard deviation of 8.90 ms, however no pattern could be observed between input payload and command execution time.

Card	Average timing (in ms)	Standard deviation (in ms)
Card A	224,26	3,89
Card B	237,98	0,76
Card C	205,48	5,21
Card D	187,02	8,90
Card E	175,22	0,70
Card F	269,08	5,85

Table 11. GENERATE AC fuzzing timing

In terms of results, no obvious and repeatable problems were determined — most often, the input would mutate into becoming unsuitable for use and error responses would emerge. The exact type of responses varied by card, however "invalid precondition" (`0x69 0x85`) and "invalid parameters" (`0x68 0x88`) were most common. Observing the timing information for fuzzing also showed consistency and no input which causes abnormally long processing times could be isolated.

5.3. Summary on tested cards

On first tests, cards C and D did not work as contactless cards — no MasterCard application was present in AFL. After using aforementioned cards in chip and pin flow, the contactless mode was enabled.

Coupled with information regarding exposed tags through `GET DATA` commands (see section 5.2.1), Cards B and D show the most similarity in behaviour and features.

Card F stands out by only exploding bare minimum set of commands to complete the payment flows and for this reason has the smallest attack surface.

Potential vulnerabilities Card C is unique in supporting Magnetic Stripe mode payments among the tested cards, using a very small pool of possible UNs. While payment solutions in Estonia do not support this legacy mode any longer, this might not be true in other countries. Card E is the only tested card to support relay resistance protocol — this was indicated in AIP response and also in presence of `EXCHANGE RELAY RESISTANCE (0x80 0xEA)` command. While it is unknown how common the support for relay resistance is in PoSs deployed in Estonia, it nevertheless suggests other cards may be susceptible to relay attacks.

The summary of card features can be found in Table 12.

	Card A	Card B	Card C	Card D	Card E	Card F
Supported logical channels	0-3	0-3	0-3	0-3	—	—
Magnetic stripe mode	—	—	yes, UN size 10^2	—	—	—
Relay attack protection	—	—	—	—	yes	—
Log format description	present	—	present	—	—	present
Transaction log	10 entries	10 entries	10 entries	10 entries	—	10 entries
ICC key size	1152 bits	896 bits	1024 bits	896 bits	896 bits	1408 bits

Table 12. Summary of test card features

6. Conclusion and future work

This thesis stated three main goals: 1) test Estonian contactless bank cards for vulnerabilities; 2) create a systematic way of testing EMV Contactless bank cards for anomalies; 3) test the method out on real bank cards. The test subjects in service of those goals were publicly obtainable Estonian bank cards. The thesis also imposed a restriction of performing all analysis of the cards in offline manner, without interacting with production bank networks. This was chosen so that the conducted research would affect only the author's test cards and no negative effects would manifest on real bank systems.

The systematic testing of the cards was done in several steps: determining the initial starting points for further steps, mapping the visible command space, determining valid parameters for unknown instructions, determining correct payload length for unknown commands, selecting candidate commands for fuzzing and finally performing randomized or mutation fuzz testing on selection of commands. Many of these steps lack pre-existing tooling, which led the author to create their own scripts to accomplish these results. The software is open source and has been made public on the author's GitHub page.

The testing showed mixed results — no overt vulnerabilities were found, however it was possible to change the internal state in one of the test cards through randomized fuzzing. The change itself was not hugely significant — responding with incorrect error codes, but it showed a flaw somewhere in the firmware. The card continued to behave normally when used at payment terminals, both in contact and contactless means. Another observation from testing is that many of the cards exposed more functionality (logical channels, leftover commands) or information that is required for their main purpose — completing payments. Finding this problem shows that the method works in uncovering unforeseen problems in card operation.

Academic research into contactless bank card security is scarce, however some vulnerabilities have been uncovered in the past decade. As all the test cards for this thesis were using the MasterCard payment network, this thesis was unable to test the vulnerabilities against Visa cards. One of the issues, pre-play attack against MasterCard cards, was successfully replicated on one of the test cards. The vulnerability works with the legacy magnetic stripe

mode, which needs to be supported by both card and PoS terminal. In certain markets, this would cause concern, however Estonia is using chip and PIN (EMV mode) payments since the introduction of swipe cards and the pre-play attack is therefore not effective. All but one of the test cards lacked protection against another well known vulnerability — relay attacks. Several papers have been written on this topic, indicating interest of security researcher community.

It is the author's opinion that future research in this area will require good cooperation with banks issuing the cards, which will allow for deeper and more intrusive experimentation. The entire payment process should be looked at — from the card itself, the PoS terminals and ATMs and local payment providers (such as Nets Estonia). The lack of access to the card internals means most of the testing is done blindly and any support from the manufacturer or the issuer would help future researchers greatly.

References

[1] MasterCard. (May 3, 2012). New MasterCard Advisors Study on Contactless Payments Shows Almost 30% Lift in Total Spend Within First Year of Adoption, [Online]. Available: https://newsroom.mastercard.com/press-releases/new-mastercard-advisors-study-on-contactless-payments-shows-almost-30-lift-in-total-spend-within-first-year-of-adoption/ (visited on Dec. 2, 2018).

[2] E. Pank, *Maksekeskkonna ülevaade 2018*, Mar. 2018. [Online]. Available: https://www.eestipank.ee/sites/eestipank.ee/files/files/Maksesysteemid/et/ep_mky_2018_est_0.pdf.

[3] M. Savage, "Why Sweden is close to becoming a cashless economy", Sep. 12, 2017. [Online]. Available: https://www.bbc.com/news/business-41095004 (visited on Nov. 10, 2018).

[4] (Mar. 5, 2013). Critical Infrastructure Sectors, [Online]. Available: https://www.dhs.gov/critical-infrastructure-sectors (visited on Nov. 10, 2018).

[5] C. for the Protection of National Infrastructure. (). Critical National Infrastructure, [Online]. Available: https://www.cpni.gov.uk/critical-national-infrastructure-0 (visited on Nov. 10, 2018).

[6] *Critical Infrastructure Resilience Strategy*. Canberra: Australian Government, 2010, OCLC: 652617702, ISBN: 978-1-921725-25-8.

[7] R. of Estonia, *Emergency Act*, Jul. 1, 2018. [Online]. Available: https://www.riigiteataja.ee/en/eli/ee/525062018014/consolide/current (visited on Nov. 10, 2018).

[8] "EMVCo Members", *EMVCo*, [Online]. Available: https://www.emvco.com/about/emvco-members/ (visited on Dec. 1, 2017).

[9] (). Worldwide EMV® Deployment Statistics, [Online]. Available: https://www.emvco.com/about/deployment-statistics/ (visited on Oct. 20, 2018).

[10] EMVCo. (2018). Contact, [Online]. Available: https://www.emvco.com/emv-technologies/contact/ (visited on Oct. 6, 2018).

[11] “Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange”, International Organization for Standardization, Geneva, CH, Standard, 1995.

[12] “Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange”, International Organization for Standardization, Geneva, CH, Standard, 2005.

[13] EMVCo, *A Guide to EMV Chip Technology*, Nov. 2014. [Online]. Available: https://www.emvco.com/wp-content/uploads/2017/05/A_Guide_to_EMV_Chip_Technology_v2.0_20141120122132753.pdf (visited on Nov. 11, 2018).

[14] ——, (May 2018). Contactless, [Online]. Available: https://www.emvco.com/emv-technologies/contactless/ (visited on Oct. 6, 2018).

[15] “News Pocopay and Nets Estonia Embark on a New Contactless Payment Journey”, [Online]. Available: https://www.nets.eu/Media-and-press/news/Pages/An-Estonian-company-Pocopay-is-a-new-player-in-the-financial-market-and-certainly-one-to-look-out-for-.aspx (visited on Dec. 1, 2017).

[16] “LHV: Viipemakse Terviklahendus”, [Online]. Available: http://www.pangaliit.ee/et/kasulik/innovatsioonipreemia-2017/87-uncatecorized/409-lhv-viipemakse-terviklahendus (visited on Dec. 1, 2017).

[17] Swedbank. (Aug. 22, 2016). Swedbank alustas puutevaba tehnoloogiaga ISIC Tudengikaartide väljastamist, [Online]. Available: https://www.swedbank.ee/private/home/more/newsandblog?20160822111406510&language=EST# (visited on Oct. 6, 2018).

[18] ——, (2018). Mobiilsed viipemaksed - Swedbank, [Online]. Available: https://www.swedbank.ee/private/d2d/mobile/contactless (visited on Oct. 6, 2018).

[19] “Deebet-Viipekaardiga Saab Maksta Kiirelt Ja Mugavalt | SEB”, [Online]. Available: https://www.seb.ee/viipekaart (visited on Dec. 1, 2017).

[20] SEB. (2018). ISIC ja ITIC viipekaardid | SEB, [Online]. Available: https://www.seb.ee/igapaevapangandus/deebetkaardid/isic-ja-itic-kaardid (visited on Oct. 6, 2018).

[21] “Coop Mastercard Deebetkaart | Coop Pank”, [Online]. Available: https://www.cooppank.ee/deebetkaart (visited on Dec. 1, 2017).

[22] “16. Oktoobrist on Viipemakse Limiit 25 Eurot | Luminor”, [Online]. Available: https://www.luminor.ee/ee/16-oktoobrist-viipemakse-limiit-25-eurot (visited on Dec. 1, 2017).

[23] Luminor. (2018). Mastercard Debit, [Online]. Available: https://www.luminor.ee/ee/era/mastercard-debit (visited on Oct. 6, 2018).

[24] H. Lõugas, “Telia ja Kaubamaja uues äpis on NFC makse, aga mitte selline, mida me iga päev kasutame”, *Geenius*, Jan. 1, 2018. [Online]. Available: https://geenius.ee/uudis/telia-ja-kaubamaja-uues-apis-nfc-makse-aga-mitte-selline-mida-iga-paev-kasutame/ (visited on Oct. 6, 2018).

[25] *Security oriented fuzzer with powerful analysis options. Supports evolutionary, feedback-driven fuzzing based on code coverage (software- and hardware-based): Google/honggfuzz*, Google, Jan. 2019.

[26] lcamtuf, *American fuzzy lop*. [Online]. Available: http://lcamtuf.coredump.cx/afl/ (visited on Oct. 20, 2018).

[27] N. Semiconductors, *Jcop 3 emv p60 security target lite*, English, May 10, 2016.

[28] M. Emms and A. van Moorsel, “Practical attack on contactless payment cards”, 2011. [Online]. Available: https://openlab.ncl.ac.uk/bhci-securityprivacy/files/2015/01/Emms.pdf.

[29] B. Cohen. (Mar. 23, 2012). Millions of Barclays card users exposed to fraud, [Online]. Available: https://www.channel4.com/news/millions-of-barclays-card-users-exposed-to-fraud (visited on Oct. 6, 2018).

[30] R. Lifchitz–BT, “Hacking the nfc credit cards for fun and debit”, 2012.

[31] M. Emms, B. Arief, N. Little, and A. van Moorsel, “Risks of Offline Verify PIN on Contactless Cards”, in *Financial Cryptography and Data Security*, A.-R. Sadeghi, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 313–321, ISBN: 978-3-642-39884-1.

[32] M. Roland and J. Langer, "Cloning Credit Cards: A Combined Pre-play and Downgrade Attack on EMV Contactless", in *Proceedings of the 7th USENIX Conference on Offensive Technologies*, ser. WOOT'13, Washigton, D.C.: USENIX Association, 2013, pp. 6–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=2534748.2534756.

[33] MasterCard, "Transaction Processing Rules", p. 318, Jun. 28, 2018. [Online]. Available: https://www.mastercard.us/content/dam/mccom/global/documents/transaction-processing-rules.pdf.

[34] PayPass, "PayPass—M/Chip Requirements", p. 84, Apr. 10, 2014.

[35] M. Emms, B. Arief, L. Freitas, J. Hannon, and A. van Moorsel, "Harvesting High Value Foreign Currency Transactions from EMV Contactless Credit Cards Without the PIN", in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14, Scottsdale, Arizona, USA: ACM, 2014, pp. 716–726, ISBN: 978-1-4503-2957-6. DOI: 10.1145/2660267.2660312. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660312.

[36] G. Hancke, *A Practical Relay Attack on ISO 14443 Proximity Cards*. 2005.

[37] M. Roland, J. Langer, and J. Scharinger, "Applying relay attacks to Google Wallet", in *2013 5th International Workshop on Near Field Communication, NFC 2013*, Feb. 5, 2013, pp. 1–6, ISBN: 978-1-4673-4837-9. DOI: 10.1109/NFC.2013.6482441.

[38] J. van den Breekel, N. Zannone, E. Poll, J. de Ruiter, S. Hegt, T. Timmerman, and K. Netherlands, "A security evaluation and proof-of-concept relay attack on Dutch EMV contactless transactions", Master's thesis, Technische Universiteit Eindhoven, 2014.

[39] M. Nemec, M. Sys, P. Svenda, D. Klinec, and V. Matyas, "The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli", in *24th ACM Conference on Computer and Communications Security (CCS'2017)*, ACM, 2017, pp. 1631–1648, ISBN: 978-1-4503-4946-8.

[40] I. Fratric. (Sep. 21, 2017). Project Zero: The Great DOM Fuzz-off of 2017, [Online]. Available: https://googleprojectzero.blogspot.com/2017/09/the-great-dom-fuzz-off-of-2017.html (visited on Oct. 6, 2018).

[41] P. Fillmore, "Crash and Pay: Owning and Cloning Payment Devices", *blackhat 2015*, 2015. [Online]. Available: https://www.blackhat.com/docs/us-15/materials/us-15-Fillmore-Crash-Pay-How-To-Own-And-Clone-Contactless-Payment-Devices.pdf.

[42] M. Turkia. (Jun. 12, 2014). Pen Testing Payment Terminals: A Step-by-Step How-To Guide, [Online]. Available: https://pen-testing.sans.org/blog/2014/06/12/pen-testing-payment-terminals-a-step-by-step-how-to-guide (visited on Oct. 6, 2018).

[43] P. Osuch. (Nov. 11, 2015). EMV Protocol Fuzzer, [Online]. Available: https://labs.mwrinfosecurity.com/blog/emv-protocol-fuzzer/ (visited on Oct. 20, 2018).

[44] (Nov. 22, 2017). Emv first part of commands by merlokk · Pull Request #465 · Proxmark/proxmark3, [Online]. Available: https://github.com/Proxmark/proxmark3/pull/465 (visited on Oct. 6, 2018).

[45] (). Cardpeek, [Online]. Available: http://pannetrat.com/Cardpeek/ (visited on Oct. 6, 2018).

[46] *A Python module for interacting with smart cards. Contribute to mit-ll/LL-Smartcard development by creating an account on GitHub*, MIT Lincoln Laboratory, Sep. 14, 2018. [Online]. Available: https://github.com/mit-ll/LL-Smartcard (visited on Oct. 6, 2018).

[47] J. Zweng, *Reads EMV cards, extract public keys, checks them for ROCA vulnerability (https://crocs.fi.muni.cz/public/papers/rsa_ccs17). Uses code from https://github.com/devnied/EMV-NFC-Paycard-Enrollment for.*. Apr. 12, 2018. [Online]. Available: https://github.com/johnzweng/android-emv-key-test (visited on Oct. 6, 2018).

[48] P. Fillmore, *Tool for printing, dumping and testing keys stored on an EMV Chip Card.: Peterfillmore/emvkeytool*, Sep. 6, 2018. [Online]. Available: https://github.com/peterfillmore/emvkeytool (visited on Oct. 6, 2018).

[49] *ROCA: Infineon RSA key vulnerability. Contribute to crocs-muni/roca development by creating an account on GitHub*, CRoCS, Oct. 2, 2018. [Online]. Available: https://github.com/crocs-muni/roca (visited on Oct. 6, 2018).

[50] M. Kysel, *This Android app collects Mag-Stripe data and CVC3 codes from PayPass cards and emulates that information. : MatusKysel/EMVemulator*, Sep. 24, 2018. [Online]. Available: https://github.com/MatusKysel/EMVemulator (visited on Oct. 6, 2018).

[51] Jackson, *Clone and emulate PayPass-enabled MasterCard credit cards for contactless transactions: Jthuraisamy/MasterTap*, Apr. 7, 2018. [Online]. Available: https://github.com/jthuraisamy/MasterTap (visited on Oct. 6, 2018).

[52] (Aug. 31, 2015). ATM Shimming and The Death of EMV 2, [Online]. Available: https://www.finextra.com/blogposting/11483/atm-shimming-and-the-death-of-emv-2 (visited on Oct. 20, 2018).

[53] J. says. (Sep. 30, 2017). NFC – Contactless Cards: Brute Forcing Processing Options, [Online]. Available: https://salmg.net/2017/09/29/nfc-contactless-cards-brute-forcing-processing-options/ (visited on Oct. 20, 2018).

[54] "EMV Contactless Specifications for Payment Systems", p. 589, 2016.

[55] M. Romulus, *Mromulus/emv-tool*, Dec. 8, 2018. [Online]. Available: https://github.com/mromulus/emv-tool (visited on Dec. 10, 2018).

[56] EFTlab. (). Complete list of EMV & NFC tags, [Online]. Available: https://www.eftlab.co.uk/index.php/site-map/knowledge-base/145-emv-nfc-tags (visited on Oct. 6, 2018).

[57] *A pure-python fully automated and unattended fuzzing framework.: OpenRCE/sulley*, OpenRCE, Oct. 11, 2018. [Online]. Available: https://github.com/OpenRCE/sulley (visited on Oct. 20, 2018).

[58] *An automated NFC fuzzing framework for Android devices.: Mit-ll/LL-Fuzzer*, MIT Lincoln Laboratory, Sep. 28, 2018. [Online]. Available: https://github.com/mit-ll/LL-Fuzzer (visited on Oct. 20, 2018).

[59] GlobalSign. (). 1024 bit Public and Private Keys, [Online]. Available: https://www.globalsign.com/en/ssl-information-center/1024-bit-public-and-private-keys/ (visited on Oct. 6, 2018).

[60] L. Constantin. (2015-01-29T11:18-05:00). The end for 1024-bit SSL certificates is near, as Mozilla kills a few more, [Online]. Available: https://www.computerworld.com/article/2877654/the-end-for-1024-bit-ssl-certificates-is-near-as-mozilla-kills-a-few-more.html (visited on Oct. 6, 2018).

[61] D. Giry. (Jun. 10, 2018). Keylength - Cryptographic Key Length Recommendation, [Online]. Available: https://www.keylength.com/ (visited on Oct. 6, 2018).

[62] (Jun. 5, 2016). Smartcard - Get Data APDU command different tags and response format, [Online]. Available: https://stackoverflow.com/questions/37640130/get-data-apdu-command-different-tags-and-response-format (visited on Nov. 22, 2018).

[63] B. Reding, "GlobalPlatform Card Specification 2.2.0.7", p. 375, 2006.

[64] EMVCo, *EMV Integrated Circuit Card Specifications for Payment Systems Book 3*. 2004, vol. 4.1.

[65] "Identification cards – Integrated circuit cards – Part 9: Commands for card management", International Organization for Standardization, Geneva, CH, Standard, Dec. 2017.

Appendices

A. GPO example from Proxmark3

```
proxmark3> script run emv-test
--- Executing: emv-test.lua, args ''
119E6093
PSE-s
>>>> 00 a4 04 00 0e 32 50 41 59 2e 53 59 53 2e 44 44 46 30 31 00
<<<< 0a 00 6f 3d 84 0e 32 50 41 59 2e 53 59 53 2e 44 44 46 30 31 a5 2b 5f 2d
    04 65 74 65 6e bf 0c 21 61 1f 4f 07 a0 00 00 00 04 10 10 50 0a 4d 61 73
    74 65 72 43 61 72 64 87 01 01 9f 0a 04 00 01 01 01 90 00
APDU response status: 9000 - Command successfully executed (OK).
------------------- TLV decoded --------------------
-- a[00] 'Unknown ???':
--6f[3d] 'File Control Information (FCI) Template':
  --84[0e] 'Dedicated File (DF) Name':
    00: 32 50 41 59 2e 53 59 53 2e 44 44 46 30 31       |2PAY.SYS.DDF01
  --a5[2b] 'File Control Information (FCI) Proprietary Template':
    --5f2d[04] 'Language Preference':       String value 'eten'
      00: 65 74 65 6e             |eten
    --bf0c[21] 'File Control Information (FCI) Issuer Discretionary Data':
      --61[1f] 'Application Template':
        --4f[07] 'Application Dedicated File (ADF) Name':
          00: a0 00 00 00 04 10 10          |.......
        --50[0a] 'Application Label':   String value 'MasterCard'
          00: 4d 61 73 74 65 72 43 61 72 64       |MasterCard
        --87[01] 'Application Priority Indicator':
          00: 01                |.
        --9f0a[04] 'Application Selection Registered Proprietary Data':
          00: 00 01 01 01             |....
Selecting mastercard AID
>>>> 00 a4 04 00 07 a0 00 00 00 04 10 10 00
<<<< 0a 00 6f 35 84 07 a0 00 00 00 04 10 10 a5 2a 50 0a 4d 61 73 74 65 72 43
    61 72 64 87 01 01 5f 2d 04 65 74 65 6e bf 0c 11 9f 6e 07 02 33 00 00 30
    30 00 9f 0a 04 00 01 01 01 90 00
APDU response status: 9000 - Command successfully executed (OK).
------------------- TLV decoded --------------------
```

```
── a[00] 'Unknown ???':
──6f[35] 'File Control Information (FCI) Template':
  ──84[07] 'Dedicated File (DF) Name':
    00: a0 00 00 00 04 10 10          |.......
  ──a5[2a] 'File Control Information (FCI) Proprietary Template':
    ──50[0a] 'Application Label':    String value 'MasterCard'
      00: 4d 61 73 74 65 72 43 61 72 64       |MasterCard
    ──87[01] 'Application Priority Indicator':
      00: 01                |.
    ──5f2d[04] 'Language Preference':       String value 'eten'
      00: 65 74 65 6e             |eten
    ──bf0c[11] 'File Control Information (FCI) Issuer Discretionary Data':
      ──9f6e[07] 'Unknown ???':
        00: 02 33 00 00 30 30 00          |.3..00.
      ──9f0a[04] 'Application Selection Registered Proprietary Data':
        00: 00 01 01 01             |....
Get processing options
>>>>[keep TLV] 80 a8 00 00 02 83 00 00
<<<< fa 00 01
APDU response: 00 01 —
```

B. Initial target APDU for mutation fuzzing

0x80 0xAE (INS CLA): `GENERATE AC`

0x90 0x00 (P1, P2): ARQC for CDA

0x00 0x00 0x00 0x00 0x00 0x01: Authorized amount (0.01 €)

0x00 0x00 0x00 0x00 0x00 0x00: Other amount (0.00 €)

0x02 0x33: Terminal country code (Estonia)

0x80 0x00 0x00 0x00 0x00 0x80: TVR result

0x09 0x78: Transaction currency code (EUR)

0x18 0x03 0x25: Transaction date (25.03.2018)

0x00: Transaction type (default)

0x00 0x00 0x00 0x00: Unpredictable number

0x21: Terminal type

0x00 0x00: Data authentication code

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00: ICC dynamic number

0x1F 0x03 0x02: Cardholder Verification results

0x15 0x50 0x00: Transaction time

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00: Merchant Custom Data