# A Security Evaluation and Proof-of-Concept Relay Attack on Dutch EMV Contactless Transactions

Jordi van den Breekel
j.m.v.d.breekel@outlook.com

Under supervision of:

Dr. Nicola Zannone
Security Group
Computer Science and Engineering
University of Technology Eindhoven

Dr. Erik Poll
Joeri de Ruiter, MSc
Digital Security Group
Institute for Computing and Information Sciences
Radboud University Nijmegen

Stan Hegt, MSc
Thijs Timmerman, MSc
Information Protection Services
KPMG Netherlands

October, 2014

# Abstract

Dutch banks introduced contactless payments in April 2014, and have been promoting the use of contactless cards since then. Contactless payments are based on the EMV specification. EMV is the worldwide standard for contact and contactless transactions. EMV Contact is a well-researched field and many vulnerabilities have been found. Although EMV Contactless is newer and less researched, a few vulnerabilities have already been identified. All known EMV Contactless vulnerabilities exist in legacy modes that EMV provides. These modes, however, are not supported in the Netherlands and thus these vulnerabilities are not applicable to Dutch cards or terminals.

We present the first vulnerabilities in EMV Contactless that are applicable to Dutch cards and terminals. We show that a relay attack can be performed with very limited resources and widely available off-the-shelf hardware. Our proof-of-concept relay attack proves that a criminal can pay at a Point-of-Sale terminal, using the card inside a wallet of a victim, while the victim is arbitrary far away from the terminal. All Dutch cards are vulnerable to our proof-of-concept relay attack. Furthermore, we identified other vulnerabilities in Knab bank and Vodafone cards, and in the most commonly used type of Point-of-Sale terminals in the Netherlands. For instance, Maestro cards issued by Knab bank have a vulnerability which concerns the distribution process of the cards. The Visa cards issued by Vodafone break the EMV security requirements by using secret keys that are not unique. The most common Point-of-Sale terminals in the Netherlands are vulnerable to a Denial-of-Service attack, which presumably is the result of a buffer overflow.

Our findings have serious implications for contactless transactions in the Netherlands. Indeed, contactless transactions will not be widely accepted by customers if they are not confident about the security of contactless cards, and banks can suffer significant reputational damage. Based on the relay attack and the other vulnerabilities, we have identified a number of realistic attack scenarios in which customers and banks could suffer significant financial loss.

Atos, Visa and Knab bank have been notified about the found vulnerabilities in the Point-of-Sale terminal, the Visa cards and the Knab cards, respectively. Atos and Visa acknowledged that future versions of their terminals and cards will be adjusted to address the vulnerabilities. The 'Betaalvereniging' also acknowledged that Knab bank is looking into the found vulnerability.

# Contents

**Abstract** i

**List of Figures** ix

**List of Tables** xi

**Glossary** xiii

**1 Introduction** 1
- 1.1 Motivations . . . 1
- 1.2 Research Questions . . . 2
- 1.3 Contribution . . . 3
- 1.4 Scope of the Research . . . 4
- 1.5 Outline . . . 4

**2 Background** 7
- 2.1 EMV Contactless . . . 7
- 2.2 EMV in the Netherlands . . . 7
- 2.3 EMV Contact Specifications . . . 10
  - 2.3.1 Transaction Flow . . . 11
  - 2.3.2 Initialization Phase . . . 11
  - 2.3.3 Card Authentication Phase . . . 13
  - 2.3.4 Cardholder Verification Phase . . . 15
  - 2.3.5 Transaction Phase . . . 17
  - 2.3.6 Cryptogram Generation . . . 20

**3 EMV Contactless Specifications** 21
- 3.1 EMV Contactless Specification Documents . . . 21
- 3.2 Contactless Interface . . . 23
- 3.3 EMV Contactless Transactions . . . 23
- 3.4 MasterCard PayPass Specifications . . . 26
  - 3.4.1 MasterCard Contactless Transaction Initialization . . . 27
  - 3.4.2 MasterCard Contactless Mag-Stripe Mode . . . 28
  - 3.4.3 MasterCard Contactless EMV Mode . . . 29

3.5 Visa payWave Specifications . . . . . 34
3.5.1 Contactless qVSDC Mode . . . . . 35
3.5.2 Contactless Mag-Stripe Mode . . . . . 35
3.5.3 Contactless VSDC Mode . . . . . 36
3.5.4 Contactless Not Supported . . . . . 37

**4 Related Work** **39**
4.1 EMV Contact Attacks . . . . . 39
4.1.1 Cloning Cards with Static Data Authentication . . . . . 39
4.1.2 Faking Transactions with Dynamic Data Authentication . 40
4.1.3 Eavesdropping . . . . . 41
4.1.4 PIN Bypass . . . . . 42
4.1.5 Relay Attack . . . . . 42
4.2 EMV Contactless Attacks . . . . . 43
4.2.1 EMV Contact Attacks on EMV Contactless . . . . . 43
4.2.2 Pre-play & Downgrade Attack on Mag-Stripe Mode . . . . 44
4.2.3 Offline PIN Verify Attack . . . . . 45
4.2.4 Harvesting High Value Foreign Currency Transactions . . . 45
4.2.5 Relay Attack with Special Hardware . . . . . 49
4.2.6 Relay Attack with Phones . . . . . 50

**5 Dutch EMV Contactless System** **53**
5.1 Dutch Implementation of EMV Contactless . . . . . 53
5.1.1 Methodology . . . . . 53
5.1.2 Maestro Card Configuration . . . . . 55
5.1.2.1 Parameterizations in Card . . . . . 55
5.1.2.2 Card Differences . . . . . 56
5.1.2.3 Card Behavior . . . . . 57
5.1.3 Visa Cards Configuration . . . . . 60
5.1.4 Overview . . . . . 61
5.2 Applicability of Known Attacks on Dutch EMV Contactless . . . 61
5.3 Formal Analysis . . . . . 64

**6 Proof-of-Concept Relay Attack** **67**
6.1 Relay Attack . . . . . 67
6.2 Methodology . . . . . 69
6.3 Equipment . . . . . 70
6.3.1 POS Terminal . . . . . 70
6.3.2 Relay Device . . . . . 71
6.3.3 Mole Device . . . . . 72
6.3.4 Smart Card . . . . . 72
6.4 Relay Setups . . . . . 72
6.4.1 Basic Relay Setup . . . . . 73

6.4.2 Preloaded, Transaction Time Optimized Setup . . . . . 74
6.4.3 Preloaded, Coupling Time Optimized Setup . . . . . 77

**7 Relay Setup Performance 81**
7.1 Timing Measurements Overview . . . . . 81
7.2 Card Timings . . . . . 82
7.3 POS Terminal Timings . . . . . 86
7.4 Relay Device Timings . . . . . 87
7.5 Mole Device Timings . . . . . 88
7.6 Network Delay . . . . . 89
7.7 Relay Setup Timings . . . . . 90
7.7.1 Basic Relay Setup . . . . . 91
7.7.2 Transaction Time Optimized, Preloaded, Relay Setup . . . 91
7.7.3 Coupling Time Optimized, Preloaded, Relay Setup . . . . 94

**8 Other Vulnerabilities 97**
8.1 Denial-of-Service on Terminal . . . . . 97
8.2 Re-use of Secret Symmetric Keys in Multiple Applications . . . . 98
8.3 Weak Generation of Cryptogram . . . . . 99
8.4 Offline PIN Verification . . . . . 100

**9 Attack Scenarios 103**
9.1 Pickpocketing . . . . . 103
9.1.1 Without PIN Knowledge . . . . . 104
9.1.2 With PIN Knowledge . . . . . 104
9.2 Denial-of-Service . . . . . 108
9.2.1 Denial-of-Service on Terminals . . . . . 108
9.2.2 Exploiting a Potential Buffer Overflow . . . . . 109
9.3 Counterfeit POS Terminals . . . . . 110
9.4 Malware on Mobile Phone . . . . . 110
9.5 Envelope Fraud . . . . . 110
9.6 Attacks on PIN Verification . . . . . 111

**10 Countermeasures and Recommendations 113**
10.1 Existing Countermeasures . . . . . 113
10.1.1 Amount Limit of Contactless Without PIN . . . . . 113
10.1.2 Countermeasures to Envelope Fraud . . . . . 113
10.2 Proposed Countermeasures . . . . . 114
10.2.1 Countermeasures on Relay Attacks . . . . . 114
10.2.1.1 Timing Restriction . . . . . 114
10.2.1.2 Card Emulation Detection . . . . . 115
10.2.2 Reducing Maximum Amount Contactless With PIN . . . . 115
10.2.3 Preventing Shoulder Surfing . . . . . 117

10.2.4 Android Malware Prevention . . . . . 117
10.3 Recommendations . . . . . 117

**11 Discussion** **121**
11.1 EMV Contactless Specifications . . . . . 121
11.2 EMV Contactless Implementations . . . . . 122
11.3 Certification Process . . . . . 123
11.4 Limitations . . . . . 123
11.4.1 Reading Distance RFID Signals . . . . . 124
11.4.2 Multiple RFID Cards in One Wallet . . . . . 124
11.5 Research Timeline . . . . . 125
11.6 EMV in the media . . . . . 125

**12 Future Work and Conclusions** **127**
12.1 Future Work . . . . . 127
12.1.1 Extending the Range of Android's NFC . . . . . 127
12.1.2 Improving Relay Transaction Time . . . . . 128
12.1.3 Anti-Collision Protocol . . . . . 128
12.1.4 Research New Features . . . . . 129
12.1.5 Fuzz Testing on POS and Card . . . . . 129
12.2 Conclusions . . . . . 129

**Appendix A Maestro Type A EMV Mode Trace** **139**

**Appendix B Maestro Type B EMV Mode Trace** **147**

**Appendix C Vodafone Card App 1 Trace** **155**

**Appendix D Vodafone Card App 2 Trace** **159**

# List of Figures


2.1 PIN and cash transactions in the Netherlands . . . . . 8
2.2 (from left to right from top to bottom) The old ABN card, the Triodos card, the ING card, the new ABN card, the Knab card, the Vodafone card, and the Vodafone Radio-Frequency IDentification (RFID) sticker . . . . . 10
2.3 Message sequence chart of the initialization phase of EMV Contact 12
2.4 Message sequence chart of the DDA authentication method of EMV Contact . . . . . 14
2.5 Message sequence chart of cardholder verification with online PIN method . . . . . 16
2.6 Message sequence chart of cardholder verification with offline encrypted PIN method . . . . . 16
2.7 Message sequence chart of cardholder verification with offline plaintext PIN method . . . . . 17
2.8 Message sequence chart of a successful offline transaction phase . 19
2.9 Message sequence chart of a successful online transaction where terminal originally decided to perform an offline transaction . . . 19
2.10 Message sequence chart of a successful online transaction where terminal originally decided to perform an online transaction . . . 20

3.2 Overview of the possible transaction methods for MasterCard and Visa cards including decision making . . . . . 25
3.4 Message sequence chart of the beginning of a MasterCard contactless transaction . . . . . 28
3.5 Message sequence chart of a successful MasterCard Contactless Mag-Stripe Mode transaction . . . . . 29
3.6 Message sequence chart of a contactless transaction where the card forces an online transaction . . . . . 31
3.7 Communication sequence of an online qVSDC transaction . . . . 36
3.8 A complete online transaction performed according to the payWave Contactless Mag-Stripe Mode specifications . . . . . 36

4.1 Message Sequence Diagram PIN Bypass attack with Man-in-the-Middle (copied from [38]) . . . . . 42
4.2 Ten steps process to harvest high value TCs (copied from [8]) . . 46
4.3 Relay setup used by Kfir and Wool (copied from [32]) . . . . . 49

5.2 Trace snippet of first response of card to terminal . . . . . 55
5.4 Random reader giving warning after two invalid Personal Identification Number (PIN) tries . . . . . 64

6.1 General message sequence diagram for the Basic Relay Setup . . . 68
6.2 Our test relay setup: (from left to right) bank card, Mole Device, Relay Device, emulated Point-Of-Sale (POS) Terminal) . . . . . 69
6.4 Preloading protocol for Maestro cards . . . . . 75
6.5 Preloading protocol for Visa cards . . . . . 75
6.6 Transaction Time Optimized Relay Setup protocol for Maestro cards after preloading . . . . . 76
6.7 Transaction Time Optimized Relay Setup protocol for Visa cards after preloading . . . . . 77
6.8 Coupling Time Optimized Relay Setup protocol for Maestro cards after preloading . . . . . 79
6.9 Coupling Time Optimized Relay Setup protocol for Visa cards after preloading . . . . . 80

7.4 Transaction times for normal transactions and transaction using only the minimum set of commands . . . . . 85
7.8 Basic Relay Setup communication diagram with typical time delays 92
7.9 Transaction times per card for normal transactions, transactions with the Basic Relay Setup and transactions with the Transaction Time Optimized Relay Setup . . . . . 92
7.10 Transaction Time Optimized Relay Setup communication diagram with typical time delays . . . . . 93
7.11 Coupling times of the Coupling Time Optimized Relay Setup . . . 95
7.12 Mole Optimized Relay Setup communication diagram with Mole Device timings . . . . . 96

8.1 ATM asking to select one of the applications found on Vodafone cards . . . . . 100

9.1 An improvised camera solution used for recording PINs . . . . . 105
9.2 A very small camera used for recording PINs . . . . . 106
9.3 Still from video claiming PINs can be captured with FLIR ONE . 107
9.4 Self made pictures of POS terminal with FLIR ONE. Left we entered 1-2-3-4-5 as PIN and right we entered 6-7-8-9-0 as PIN . . . 108
9.5 An example of a phone cover that also stores bank cards . . . . . 111

10.1 Transaction receipt of contactless payment of €60 with PIN . . . 116

# List of Tables


3.1 Overview of all EMV Contact and Contactless books . . . . . 22
3.3 Market names for different systems per brand . . . . . 26

4.4 Leech to tag effort and benefit (copied from [32]) . . . . . 50

5.1 Available Dutch cards (as of July 2014) . . . . . 54
5.3 All tested Dutch EMV Contactless cards . . . . . 61

6.3 Analyzed cards . . . . . 72

7.1 Timing results of Maestro commands per smart card in [ms] . . . 83
7.2 Timing results of reading records per smart card in [ms] . . . . . 84
7.3 Timing results of Visa commands per smart card in [ms] . . . . . 84
7.5 Timing results of Nexus 7 emulating different bank cards in [ms] . 88
7.6 Timing results of Maestro commands measured with the Mole Device in [ms] . . . . . 89
7.7 Timing results of Visa commands measured with the Mole Device [ms] . . . . . 90

# Glossary

**AAC** Application Authentication Cryptogram is a cryptogram from the card for the terminal that indicates that the transaction is aborted.

**AC** Application Cryptogram is a cryptogram provided by the card as a response on a request from the terminal to perform a transaction.

**AFL** Application File Locator is a list that consists of the files and related records for the currently selected application that should be read by the terminal.

**AID** Application Identifier is an identifier to address an application in smart cards.

**AIP** Application Interchange Profile denotes the options supported by the card concerning card and issuer authentication, cardholder verification and risk management.

**ARQC** Authorization Request Cryptogram is a cryptogram from the card for issuer, forwarded by the terminal, indicating that the issuer must authorize the card to perform the transaction.

**ATC** Application Transaction Counter is a card controlled counter that increments with each transaction.

**ATM** An Automated Teller Machine is a bank controlled machine where customers can withdraw money from their account using their bank card.

**ATR** Answer To Reset is the first message sent by a smart card following a power up or a hard reset.

**CDA** Combined Dynamic Data Authentication is a method to authenticate a card together with the transaction details to a terminal with a challenge-signed response mechanism.

**CDOL** Card Risk Management Data Object List is a list that consists of the data objects that must be sent from the terminal to the card in order to perform a transaction.

**CID** Cryptogram Information Data is a parameter sent with the request and the response of an AC, to indicate whether the request or response is an AAC, TC or ARQC.

**CTQ** Card Transaction Qualifiers are placed by the issuer on a card and then contain the transaction preferences of the issuer.

**CVC** Card Verification Code is a card generated code that authenticates the card and can be verified by issuer.

**CVM** Cardholder Verification Method is a method to verify the identify of the cardholder (either online PIN, offline encrypted PIN, offline unencrypted PIN, cardholder signature or no verification).

**DDA** Dynamic Data Authentication is a method to authenticate a card to a terminal with a challenge-signed response mechanism.

**DDOL** Dynamic Data Authentication Data Object List is a DOL used for DDA.

**DOL** Data Object List is a list sent from card to terminal to specify what data objects, and in which format, the terminal needs to send to the card.

**fDDA** fast Dynamic Data Authentication is a method to perform card and transaction authentication introduced by Visa's payWave. It is faster than DDA and CDA and therefore more suitable for contactless transactions.

**HCE** Host-based Card Emulation is a technique where software emulates a smart card by directly routing the NFC signals to the processor of the device.

**MITM** A Man-In-The-Middle is a person or device that directly communicates with two devices while the devices think they are communicating directly to each other. The MITM can alter, insert or delete sent messages.

**NFC** Near Field Communication is a collection of standards and technical specifications that defines how two nearby supporting devices can communicate via inductive coupling of radio frequency fields.

**PCII** POS Cardholder Interaction Information, sent from a cardholder device to the POS terminal to indicate whether the PIN was entered correctly on the cardholder device.

**PDOL** Processing Options Data Object List is a list provided by a card that specifies which data objects the reader should include when issuing the GET PROCESSING OPTIONS command.

**PIN** A Personal Identification Number is a code that is programmed in the bank card and only known to the card holder. PINs are used to verify the identity of the cardholder.

**POS terminal** Point-Of-Sale Terminal is a terminal that can interact with bank cards in order to perform transactions.

**PPSE** The Proximity Payment System Environment is selected by the POS on the card at the beginning of each transaction. The card returns all (contactless) payment applications.

**PSE** The Payment System Environment is selected by the POS on the card at the beginning of each transaction. The card returns all (contact) payment applications.

**RFID** Radio-Frequency Identification is a technique used to read and write contactless from a reader to a RFID tag.

**SDA** Static Data Authentication is a method to authenticate a card to a terminal with static data.

**SDAD** Signed Dynamic Application Data is signed response to the challenge sent from the terminal in order to perform DDA.

**SIM** Subscriber Identity Module is a contact smart card which is used in mobile devices to authenticate subscribers.

**SSAD** Signed Static Authentication Data is the signed hash of the records indicated by the AFL, signed by the issuer of the card.

**TC** Transaction Certificate is proof from the card for the terminal that a transaction took place.

**UN** Unpredictable Number is a pseudo random number used as challenge for transactions provided by the terminal.

# Chapter 1

# Introduction

Several major banks in the Netherlands have enabled contactless payments for small transactions with NFC-supporting terminals for their customers in 2014. NFC stands for near field communication and it is the next development in payment methods after EMV Contact cards and magnet stripes. EMV is a global standard that strives for compatibility between all bank cards and terminals in the world and is defined and managed by the private corporation EMVCo. EMVCo is named after its founders Europay, MasterCard and Visa and manages the standards for both contact cards and contactless cards.

EMV Contact cards have been introduced to reduce the exponential growth of skimming damages of magnet stripes. The roll-out of contactless cards is not necessary for security reasons but research suggests that by improving user experience, customers are likely to spend 30% more money[1]. Questions arise about the security and reliability with almost every introduction of new techniques, especially if they involve financial transactions. A new step in technology must not mean a step back in security properties. Intuitively, a protocol for contactless transactions requires additional security measures than a protocol for contact chip transactions, since the contactless nature introduces new attack factors.

## 1.1 Motivations

Insecure payment systems can lead to significant financial damages. Skimming damages were estimated at €38.9 million in 2011 in the Netherlands alone[2]. With card skimming, information from the magnetic stripe is copied to clone cards and to perform transactions without presence of the original card. These

[1] http://newsroom.mastercard.com/press-releases/new-mastercard-advisors-study-on-contactless-payments-shows-almost-30-lift-in-total-spend-within-first-year-of-adoption/

[2] http://www.nvb.nl/nieuws/2012/1021/betalingsverkeer-veilig-ondanks-toename-fraude.html

skimming damages were greatly reduced when the Netherlands started using the EMV Contact chip instead of the magnetic stripe for payments. Furthermore, it became even harder for criminals to use the copied cards because banks disabled payments in several countries (that were popular with criminals to use skimmed cards) by default. Skimming damages dropped to €29 million in 2012 and to €6.8 million in 2013.

The EMV Contact chips cannot be skimmed and their introduction has practically stopped this kind of attack. However, many additional vulnerabilities of the EMV Contact chip have been found. In the beginning of EMV Contact, the information from the contact chip could still be copied to another card to create a working copy. Furthermore, the EMV Contact chip still contained the magnetic stripe data, making it also possible to create a working magnetic stripe copy. Researchers even found ways to perform transactions without having to enter the PIN code of the card. All these vulnerabilities were addressed after they became known.

In April 2014, EMV Contactless transactions have been enabled in the Netherlands. EMV Contactless is relatively new, however, the first vulnerabilities of EMV Contactless have already been identified and reported in the UK. In addition, criminals have shown to be able to use very sophisticated methods and specialized hardware to commit fraud with the vulnerabilities of the magnetic stripe and the EMV Contact cards. Therefore, it is very important to have a security assessment on Dutch EMV Contactless cards, so that banks can take additional measures to prevent large scale fraud, rather than fixing it when it is already too late.

## 1.2 Research Questions

The main research question to be answered is:

*To what extent do exploitable vulnerabilities exist in the protocol specifications of EMV Contactless?*

The EMV specifications do not provide high level security requirements. So to further specify what is meant with *exploitable vulnerabilities*, we first define the most important security requirement for payment transactions: "*The cardholder, merchant and issuer should always agree after each transaction on the transaction's parameters including account numbers, amount, whether the transaction was accepted or rejected and which authentication and verification methods were used*". For our research, we state that if an attack can break this requirement, then an exploitable vulnerability exists in the protocol specification.

To answer this question we first need to understand how the EMV Contactless system works. However, EMV Contactless is described in 10 books [13–22]

(complimented with the four books on EMV Contact [9–12]), which comprise over 2276 pages. These specifications provide a protocol framework, with many optional modes and much freedom for the implementers. Moreover, security requirements and guidance are underspecified, and many technical details are scattered throughout the high-level overview. Due to the framework's flexibility, complexity, size, and underspecified security guidance, it is extremely difficult to assess the security of the EMV framework.

Because the EMV specifications describe a protocol suite rather than a single protocol, it could be the case that the Dutch implementation of EMV Contactless does not have the flaws that are introduced by the specifications. Furthermore, it is possible that the Dutch implementation is not compliant with the EMV specifications so that new flaws are introduced. Therefore, after the first research question is answered, we can perform experiments and research specifically on Dutch EMV Contactless cards and Point-of-Sale terminals to answer the second research question:

*To what extent do practically exploitable vulnerabilities exist in the Dutch implementation of EMV Contactless?*

The different implementations vary between countries and between banks. The Dutch implementation indeed is significantly different from the implementation in the UK, which is the only implementation that is described to some extent. There are no descriptions available of the Dutch EMV Contactless cards and Point-of-Sale terminals, so we have to establish which modes are supported, and test and verify every possible function and feature for presence. Only when all modes and functions are known, then a thorough security assessment is possible.

## 1.3 Contribution

We have studied and researched the EMV Contactless specifications to a great extent. Furthermore, we simulated communication between contactless cards and reader to better understand the general transaction flow. We summarized the EMV specifications to a readable and understandable summary of 16 pages. Our summary provides high-level overviews and enough technical details to make a thorough assessment of the security of these specifications.

We examined all current Dutch cards and present an overview of all functions and modes that are supported. We verified that none of the known attacks are applicable to the Dutch cards. We identified several new flaws in Dutch EMV Contactless cards and in Dutch Point-of-Sale terminals. In particular, we developed a practical, low budget, proof-of-concept relay attack with off-the-shelf hardware that can be seen as an extension of previous research on relay attacks [32, 33, 39]. We show that our proof-of-concept relay attack works on all

available Dutch cards and on the most common type of Point-of-Sale terminals in the Netherlands. Furthermore, we identified several other vulnerabilities with Knab bank cards, Visa Vodafone cards, and the Atos Worldline Point-of-Sale terminal series. Based on the relay attack and the other vulnerabilities, we have identified several realistic attack scenarios.

We presented our findings to the Betaalvereniging Nederland (the Dutch Payment Association). Representatives of all major Dutch banks were present and acknowledged our identified vulnerabilities. Knab, Visa and Atos have been notified of our findings. Atos informed us that they are working to fix the identified vulnerability. Visa also acknowledged the identified vulnerability and stated that it will be fixed in future versions of the Vodafone cards. The Betaalvereniging informed us that Knab bank is looking into the matter.

## 1.4 Scope of the Research

This thesis focuses on all EMV Contactless cards available in the Netherlands. EMV Contactless implementations in mobile phones are not considered. The most common POS terminals in the Netherlands, the Atos Wordline series, are also considered.

## 1.5 Outline

The remainder of this thesis is structured as follows:

**Chapter 2** describes the background on EMV in the Netherlands, how the EMV Contact protocol suite works and the typical transaction flows.

**Chapter 3** describes the EMV Contactless standard in general and the MasterCard and Visa parts of the standard more specifically.

**Chapter 4** describes the related work on EMV Contact and EMV Contactless, including all known attacks.

**Chapter 5** describes the results of our research on the Dutch EMV Contactless cards. Among others, the applicability of the known attacks on the Dutch implementations, and the configurations and functionalities of the cards are discussed.

**Chapter 6** describes our proof-of-concept relay attack.

**Chapter 7** presents performance measurements and the timings of the relay attack.

**Chapter 8** describes all other vulnerabilities of the Dutch EMV implementations that were accidentally found during the development of the relay attack.

**Chapter 9** gives descriptions of realistic attack scenarios exploiting the relay attack and the other identified vulnerabilities.

**Chapter 10** gives an overview of the existing and proposed countermeasures and our recommendations.

**Chapter 11** gives a discussion on our findings.

**Chapter 12** gives the conclusions of our research and discusses the possible future work.

All hyperlinks given in this thesis have been checked and are working on the 25th of October 2014.

# Chapter 2

# Background

This chapter gives the background on EMV in general and in the Netherlands specifically. Section 2.1 gives some background information on EMV Contactless in general. Section 2.2 describes the background of EMV in the Netherlands. Section 2.3 gives an extensive overview of the EMV Contact specifications.

## 2.1 EMV Contactless

EMV Contactless is the latest standard of bank cards. It is not placed as the successor of EMV Contact, but as a faster alternative. The core of the protocol suite of EMV Contactless is still the same as EMV Contact but there are some differences in options, possibilities, supported features and of course a different interface. While EMV Contact uses the contact chip in bank cards, EMV Contactless uses the RFID chips available in the newer bank cards. These RFID chips can communicate and carry out transactions with Near Field Communication (NFC) enabled POS terminals.

Bank cards are typically sized according to the ISO/IEC 7810 ID-1 specifications [25]. The contact chips on these cards are based on the ISO/IEC 7816 specifications [30] and the 'contactless integrated circuit' is based on the ISO/IEC 14443 1-4 specifications [26–29].

Because the framework for EMV Contactless and EMV Contact is the same, many of the flaws found in EMV Contact might possibly also exist in EMV Contactless. Throughout this thesis we explicitly use the different terms 'EMV Contact' and 'EMV Contactless' to indicate which protocol we are discussing.

## 2.2 EMV in the Netherlands

PIN-transactions are the most popular way to pay after cash transactions in the Netherlands. The number of PIN transactions still grows with the years while the number of cash transactions decreases. If these trends continue, PIN transactions

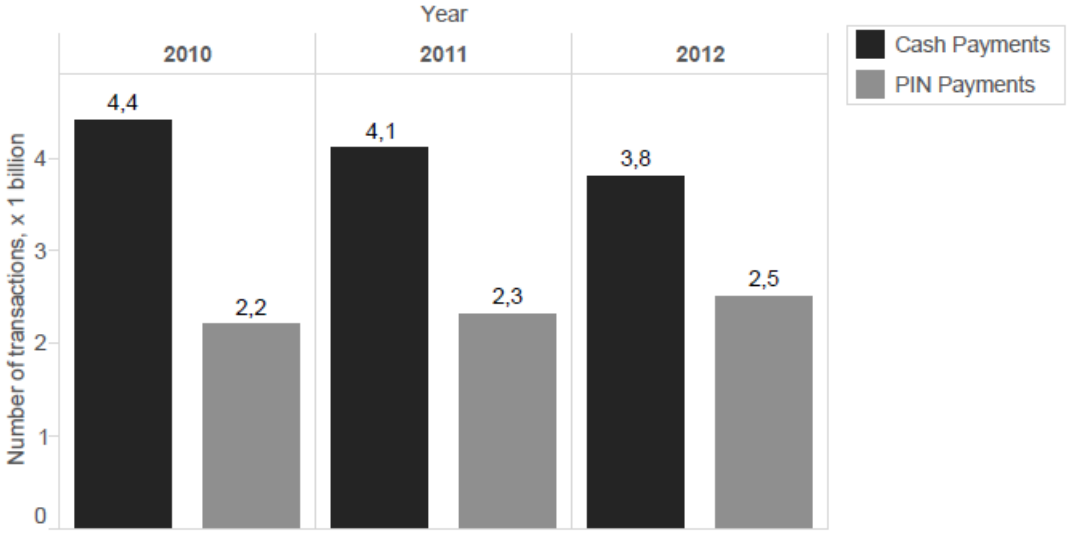


Figure 2.1: PIN and cash transactions in the Netherlands

soon will be more common than cash transactions (see Figure 2.1 for the numbers[3] in a graph).

Until 2012, magnetic stripe transactions were the default way to pay with PIN in the Netherlands. In 2012, the country switched completely to the EMV Contact chip by the name 'Het nieuwe pinnen'[4]. This transition took place two years earlier than originally planned to deal with the growth of skimming damages. In 2013 the first EMV Contactless cards were issued and contactless transactions were first enabled in 2014.

EMV Contactless transactions do not always require PIN verification and they take less time to perform than EMV Contact transactions. For these reasons, the main market targeted for contactless transactions are places where customers have little time and want to purchase small items, such as train stations or coffee shops. Especially in these circumstances customers can be convinced to quickly get a cup of coffee before their appointment begins or before their train leaves.

**Contactless Card Issuers** In the Netherlands there are four banks that issue contactless cards. ING Bank and ABN Amro bank are two of the three major banks that issue contactless cards. The third major bank of the Netherlands, Rabobank, decided to skip the contactless cards and focus on contactless transactions with mobile phones. Triodos and Knab are niche banks that also issue

[3] http://www.dnb.nl/nieuws/nieuwsoverzicht-en-archief/dnbulletin-2012/dnb274102.jsp

[4] http://www.rijksoverheid.nl/nieuws/2011/03/02/de-jager-geeft-startsein-voor-het-nieuwe-pinnen.html

contactless cards. ING Bank, ABN Amro, Triodos and Knab all issue Maestro branded contactless cards. Maestro is MasterCard's main debit card brand in the European Union.

Over more than 4.5 million contactless EMV cards are in service issued by ING Bank[5] and ABN Amro Bank. Cards issued since 2013 have contactless capabilities by default. ING and ABN Amro, however, have an opt-out policy: if customers do not want a contactless card they can request a 'traditional' EMV Contact card. By 2018 all 'old', non-contactless cards are expired and most of them will be replaced by contactless cards. This means the vast majority of the 12.5 million customers of ABN Amro and ING in the Netherlands will have contactless cards by 2018.

In addition to these four banks, there is also the telecommunication service provider Vodafone that provides contactless payment options with cards. Vodafone works together with Visa to provide their service called 'Vodafone Smartpass'. NFC-enabled and compatible mobile phones can use a special Subscriber Identity Module (SIM) together with the Vodafone app to make contactless transactions. Customers with mobile phones that are not NFC-enabled or not compatible still can make contactless transactions with a Vodafone RFID sticker. In addition, Vodafone offers a card that looks and acts like a 'normal' bank card that can be used for contact and contactless transactions.

Big differences exist between the way these contactless cards can be acquired. With ING Bank, new customers can go to an office to register and fifteen minutes later walk out with a temporary contactless card. Three working days later the long term card is sent by mail. With ABN Amro bank new customers also have to go to the office but they do not get a temporary card. After a week they also receive their contactless cards by mail. Triodos and Knab require new customers to transfer at least 1 cent from a verified bank account to the new account in order to authenticate their new customers.

Vodafone claims the sticker needs to be attached to a mobile phone[6], but states it will work even if the phone is powered off. In reality, the sticker does not have to be on a mobile device. It also works if it is not attached to anything.

Figure 2.2 shows (from left to right from top to bottom) all our tested cards: the old ABN card, the Triodos card, the ING card, the new ABN card, the Knab card, the Vodafone card, and the Vodafone RFID sticker. Interesting to notice is that there are four different physical chips issued. In particular, the ABN cards have the same physical chip, the Triodos and Knab cards have the same physical chip, but the Vodafone and ING card have chips that are not issued by other issuers. Vodafone has made a mistake with the layout of their sticker. The sticker shows the V PAY logo which indicates that the card *must* have a contact

[5] https://www.ing.nl/nieuws/nieuws_en_persberichten/2014/04/contactloos_betalen_mogelijk_voor_ing_klanten.aspx

[6] https://www.vodafone.nl/shop/mobiel/abonnement/extra-opties/apps/smartpass.shtml

Figure 2.2: (from left to right from top to bottom) The old ABN card, the Triodos card, the ING card, the new ABN card, the Knab card, the Vodafone card, and the Vodafone RFID sticker

chip[7] and that the card exclusively uses the chip and pin functionality[8], while in reality it does not have a contact chip and (as a result) does not support chip and pin.

## 2.3 EMV Contact Specifications

The EMV Contact specifications do not describe one particular protocol, but merely a protocol toolkit suite with many different options and parameterizations. Implementers can choose to some extent which options are supported and which are not, resulting in many different possible implementations. In general these specifications deal more with the functionality of the system rather than the security of the system. The main goal of the specifications is to achieve correct operation and interoperability and not 'secure transactions'.

One very important and recurring aspect in the specification is the Data Object List (DOL). A DOL is a list provided by the card that specifies the data and format it expects to receive from the terminal. This increases the flexibility and minimizes the processing within the card since it knows exactly in what format it is receiving the data.

**Key Management** All EMV Contact cards can basically be divided into two categories: cards that can perform asymmetric cryptography and cards that cannot perform asymmetric cryptography. All cards have a unique secret symmetric key $K_{IC}$ that is shared between card and issuer. These secret symmetric keys are used by the card to prove to the issuer that the card approved the transaction.

[7] `http://www.vpay.com/nl/`
[8] `http://www.vpay.com/main.html`

Issuers have public-private $(pk_I, sk_I)$ key pairs. These public keys are known to terminals. Terminals have the public keys of the issuer, and can thus check certificates that are signed by the issuer and contained in cards. Cards that can perform asymmetric cryptography also have public-private $(pk_C, sk_C)$ key pairs. The public key of the card is typically contained in a certificate stored at the card, signed by the issuer, so that the POS terminal can verify the public key of the card. The card can then prove authenticity to the POS terminal by signing data with its private key. The private key never leaves the card.

**Data Authenticity** All cards can prove authenticity of the data to the issuer by computing the hash over the data and encrypting the hash with the shared symmetric key $K_{IC}$. The terminal does not have this key and thus cannot check the authenticity of the data. Cards that support asymmetric cryptography can additionally provide digitally signed data to prove data authenticity to issuer and terminal.

### 2.3.1 Transaction Flow

Each successful transaction carried out with EMV Contact can roughly be divided in four different phases: initialization, card authentication, cardholder verification and transaction phases. In the remainder of this section, we present these phases in more detail.

### 2.3.2 Initialization Phase

The goals of the initialization phase are that the right application is selected and that all mandatory information is exchanged from card to terminal to determine how to proceed for the next steps. The message sequence chart of the initialization phase is shown in Figure 2.3.

In Figure 2.3 the flow is shown for the initialization phase. The terminal starts by requesting the Payment System Environment (PSE) from the card (message 1). The response of the card contains all the payment applications that are supported by the card (message 2). The terminal can then select one application that it wants to use on the card (message 3). The terminal then requests the processing options. The card responds with the Application Identifier (AID) that was selected (message 4). The terminal requests the processing options (message 5) and the card responds with its Application Interchange Profile (AIP) and Application File Locator (AFL) (message 6). The AIP contains the following information:

- Whether Static Data Authentication (SDA) is supported,
- Whether Dynamic Data Authentication (DDA) is supported,

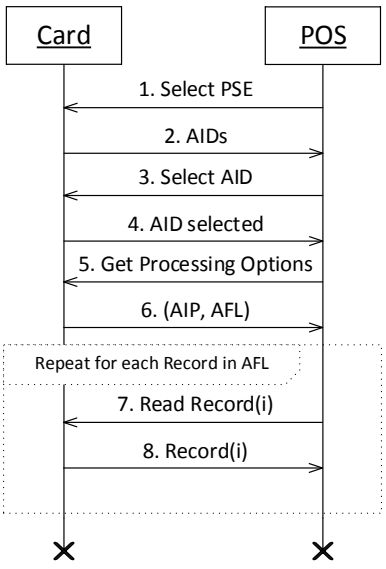


Figure 2.3: Message sequence chart of the initialization phase of EMV Contact

- Whether Combined Dynamic Data Authentication (CDA) is supported,
- Whether cardholder verification is supported,
- Whether terminal risk management is to be performed,
- Whether issuer authentication is supported.

SDA, DDA and CDA are three different methods to authenticate the card to the terminal. These options are further discussed in the Section 2.3.3.

The AFL is a list that consists of the files and related records for the currently selected application that should be read by the terminal. The terminal requests these records and the card sends them to the terminal (messages 7 and 8). These records contain at least the following mandatory information:

- Application Expiration Date,
- Application Primary Account number,
- Card Risk Management Data Object List 1 (CDOL1),
- Card Risk Management Data Object List 2 (CDOL2),
- Certification Authority Public Key Index,
- Issuer Public Key, Remainder and Exponent.

The Card Risk Management Data Object List (CDOL) specifies which data objects must be sent from the terminal to the card when the terminal issues a `GENERATE AC` command. The `GENERATE AC` command is sent by the terminal to the card to perform a transaction. If the card requests online authorization, a second `GENERATE AC` command is used by the terminal. This process is explained in detail in Section 2.3.5. Data objects specified by CDOL1 are sent with the first `GENERATE AC` command, and data objects specified by CDOL2 are sent with the second `GENERATE AC` command (if used).

For cards that support SDA this list is extended with:

- Signed Static Authentication Data (SSAD).

The SSAD is the signed hash of the records indicated by the AFL, signed by the issuer of the card. SSAD is explained more in Section 2.3.3.

For cards that support DDA or CDA this list is extended with:

- Card Public Key, Remainder and Exponent,
- Dynamic Data Authentication Data Object List (DDOL), [optional].

The DDOL is used for the `INTERNAL AUTHENTICATE` command, which is explained more in Section 2.3.3.

Now that the terminal and card agree on a selected application and the terminal has all the necessary information from the card, the protocol progresses to the card authentication phase.

### 2.3.3 Card Authentication Phase

The goal of the card authentication phase is that the card authenticates its genuineness and which bank issued the card to the terminal. The EMV Contact specifications offer three different card authentication methods. The terminal at this point knows which methods are supported by the card, and the terminal knows of course which method it supports itself. The method that is supported by both with the highest priority is selected. CDA always has the highest priority, then DDA and then SDA has the lowest priority. For all three methods, only the card is authenticated to the terminal and the terminal is not authenticated to the card. These three methods all use the so called 'offline data authentication'. They are especially useful for offline transactions, where the issuer cannot authenticate the card. This way, the terminal can authenticate the card and there is card authentication even for offline transactions. These three methods are explained in the remainder of this section.

**Static Data Authentication** SDA has the lowest priority of the three methods and is only used if no other method is jointly supported by card and terminal.

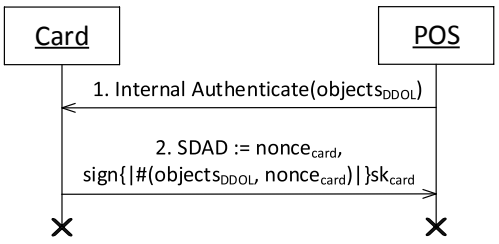


Figure 2.4: Message sequence chart of the DDA authentication method of EMV Contact

The terminal verifies if the hash received from the SSAD is the same as the hash calculated from the records mentioned by the AFL. Then the terminal verifies the sign on the hash if it was indeed signed by the legitimate issuer of the card. Now the authentication is finished and the protocol moves to the next phase.

**Dynamic Data Authentication** If DDA is selected as method, the terminal first has to check the card's public key. This public key is signed by the card issuer, whose public key is signed by a certification authority. So in order to verify the card's public key, first two signed certificates have to be verified. The message sequence chart of the DDA authentication method is shown in Figure 2.4.

Figure 2.4 shows the flow for DDA. Now that the public key of the card is verified, the terminal sends a challenge to the card to verify that the card has actually access to the corresponding private key (message 1). For this the command `INTERNAL AUTHENTICATE` is used. The data objects specified by the DDOL are included with this challenge. If no DDOL was provided by the card (since it is optional), a (terminal) default DDOL is used. A terminal generated nonce is the only mandatory data object for the `INTERNAL AUTHENTICATE` command.

The card responds by sending its Signed Dynamic Application Data (SDAD) (message 2). This SDAD includes a card generated nonce and the digital signature over the hash of the objects specified by the `INTERNAL AUTHENTICATE` command and the card generated nonce. Once the terminal has verified the hash and signature, the card is authenticated to the terminal and the protocol moves to the next phase.

**Combined Data Authentication** CDA uses asymmetric encryption not only to authenticate the card to the terminal, but also to authenticate the transaction information. The authentication of the card is combined with the authentication of the transaction details. The authentication is performed by means of a `GENERATE AC` command. For now it suffices to state that the response to this command is a card generated nonce, transaction specific data and a signature

over the hash of the data objects specified by the DDOL, the card generated nonce and transaction specific data. What data the transaction specific data consists of is explained more in-depth in Section 2.3.5.

### 2.3.4 Cardholder Verification Phase

The goal of the cardholder verification phase is to verify the identity of the cardholder, so that stolen or lost cards cannot be easily used. The card optionally provides the terminal its Cardholder Verification Method (CVM)s list with verification methods that the card supports, ordered by preference and under which conditions they are acceptable. If the card does not have a CVM list or if the list is empty, the transaction can be carried out without cardholder verification. The five different methods to verify the cardholder are:

- Online PIN verification,
- Offline encrypted PIN verification,
- Offline plaintext PIN verification,
- Cardholder signature verification,
- No cardholder verification.

The terminal selects the method most preferred by the card, that is acceptable given the circumstances and also supported by the terminal. If a method from the CVM list cannot be performed (because the circumstances are not met or the terminal does not support it) the terminal either selects the next method from the CVM list or declares the cardholder verification unsuccessful depending on what action is specified by the CVM list. In the remainder of this section the five different methods are discussed.

**Online PIN Verification** The message sequence chart of online PIN verification is shown in Figure 2.5. When the PIN is verified online it is always first entered plaintext in the keypad by the cardholder. The PIN is then encrypted in the terminal and sent to the issuer over a secure channel (message 1). This issuer then verifies the PIN and responds whether the PIN was entered correctly (message 2).

**Offline Encrypted PIN Verification** The message sequence chart of the offline encrypted PIN verification method is shown in Figure 2.6. When the PIN is verified offline the terminal first requests a nonce from the card (message 1). The card responds with a nonce (message 2). This nonce is encrypted together with the PIN (so that the ciphertext is different each time) and sent to the card (message 3). The card decrypts the ciphertext and checks the nonce and PIN

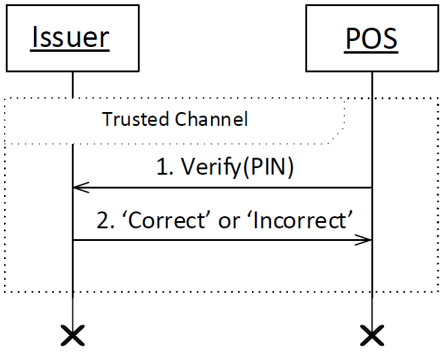


Figure 2.5: Message sequence chart of cardholder verification with online PIN method

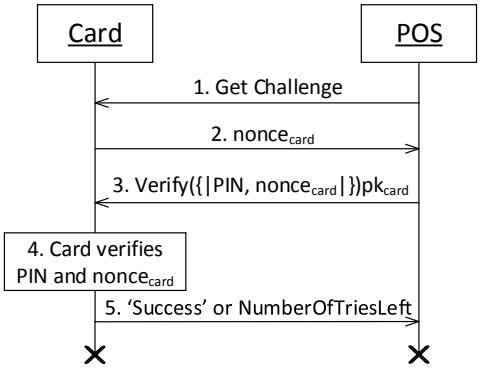


Figure 2.6: Message sequence chart of cardholder verification with offline encrypted PIN method

(action 4). If both are correct the card sends a 'Success' message. Otherwise it sends a 'Failed' message together with the number of PIN tries there are left (message 5).

**Offline Plaintext PIN Verification** The message sequence chart of the offline plaintext PIN verification method is shown in Figure 2.7. If the offline plaintext PIN verification method is use,d the PIN is sent plaintext from the terminal to the card (message 1). The card verifies this PIN (action 2) and the response is the same as with offline encrypted verification, i.e., a 'Success' is sent if the PIN is correct or a 'Failed' message together with the number of PIN tries left is sent otherwise (message 3).

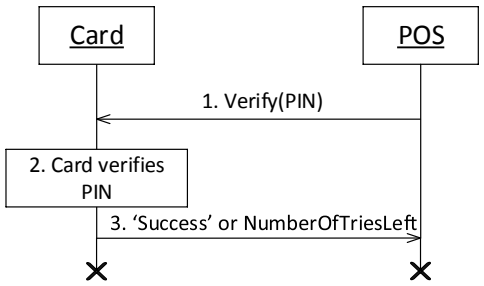


Figure 2.7: Message sequence chart of cardholder verification with offline plaintext PIN method

**Cardholder Signature Verification** The EMV specifications do not describe the cardholder written signature verification in great detail and they are not unambiguous about this method. What is described is that terminals must be attended to support signature verification and must *support* a printer. This does not necessarily mean, depending on your interpretation, that there has to be a functional printer connected. However, the printed receipt must contain a line for cardholder signature. The specifications, however, do not require the cardholder to actually sign this line on the receipt and do not require the merchant to verify this signature with the one on the card. The transaction is considered successful as soon as the terminal indicates that it supports the cardholder signature verification method.

**No Cardholder Verification** If the 'no cardholder verification' method is chosen by card and terminal, no verification of the cardholder has to take place and the transaction can be finalized.

### 2.3.5 Transaction Phase

During the last phase of the protocol, the transaction is finalized. Transactions can be performed online or offline and they can be rejected. In order to determine what is the preferred option given the transaction details and circumstances (such as amount, country, card authentication method and CVM), risk management is performed both in the terminal and in the card. We first explain some of the terms we use in this section in order to explain how the transaction phase works:

- `GENERATE AC`, a command sent from the terminal to the card to indicate that the terminal requests an Application Cryptogram (AC). Parameters of the command indicate which type of AC is requested, whether CDA is to be performed and the objects specified by CDOL1 or CDOL2,

- AC, a cryptogram from the card for the terminal which can be of three different types: Transaction Certificate (TC), Authorization Request Cryptogram (ARQC) or Application Authentication Cryptogram (AAC),
- TC, proof from the card for the terminal that a transaction took place,
- ARQC, a cryptogram from the card for the issuer, forwarded by the terminal, indicating that the issuer must authorize the card to perform the transaction. The terminal receives the approval from the issuer and forwards it to the card,
- AAC, a cryptogram from the card for the terminal that indicates that the transaction is aborted.

The response from the card always contains the following information:

- Cryptogram Information Data (CID), indicating what type of AC it is,
- Application Transaction Counter (ATC), a card controlled counter that increments with each transaction,
- AC, which should consist of the encrypted hash over the amount, country, Transaction Verification Results, Currency, Date, Transaction Type, Unpredictable Number (UN) from terminal, AIP (optionally), ATC (optionally) and SDAD (only if terminal requested CDA).

The message sequence chart of a successful offline transaction phase is shown in Figure 2.8. If the terminal decides to perform the transaction offline (action 1), it requests a TC from the card with the `GENERATE AC(TC)` command (message 2). The card can agree (action 3) with performing the transaction offline by sending a TC (message 4).

The message sequence chart of a successful online transaction phase where the terminal originally decided to perform an offline transaction is shown in Figure 2.9. If the terminal decides to perform the transaction offline (action 1 and message 2) but the card finds it necessary to perform the transaction online (action 3), the card indicates this by sending an ARQC to the terminal (message 4). The terminal forwards this ARQC to the issuer (message 5) which authorizes or rejects the transaction (message 6). If the terminal receives an authorization, it sends this to card with a second `GENERATE AC(TC)` command or with an `EXTERNAL AUTHENTICATE` command (message 7). Now the card is authorized by the issuer and sends a TC to the terminal (message 8).

If the terminal decides to perform an online transaction, the card cannot demand an offline transaction. The message sequence chart of a successful transaction phase where the terminal originally decided to perform it online is shown in Figure 2.10. If the terminal decides to perform the transaction online (action 1), it requests an ARQC from the card with the `GENERATE AC(ARQC)` command

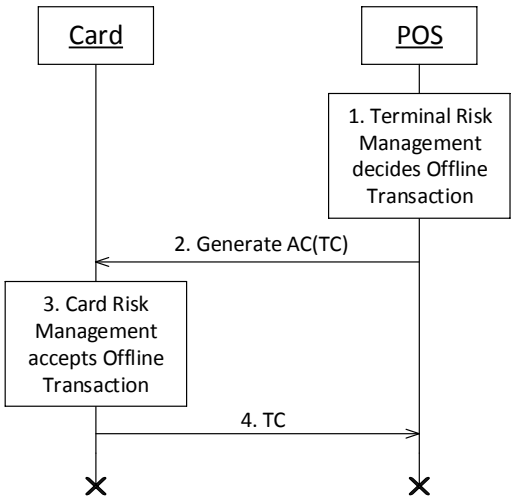


Figure 2.8: Message sequence chart of a successful offline transaction phase

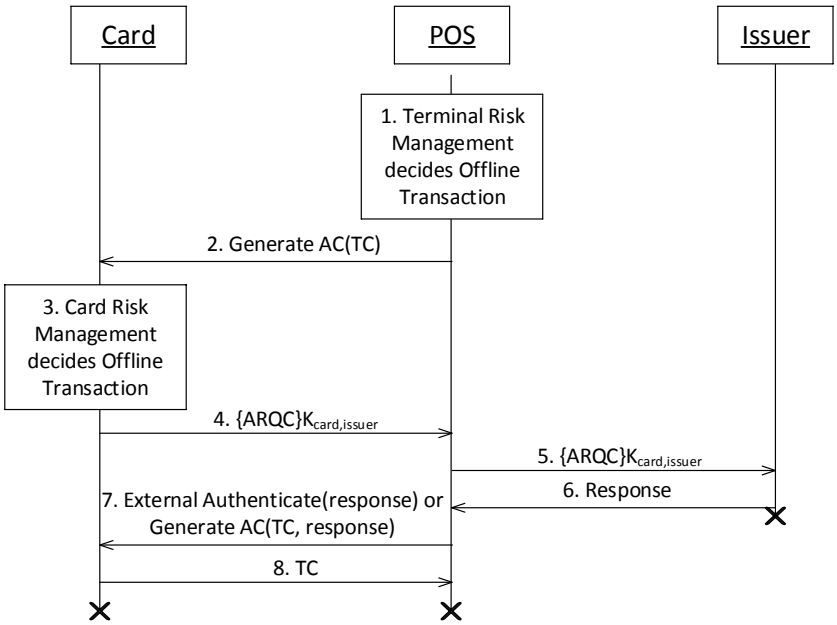


Figure 2.9: Message sequence chart of a successful online transaction where terminal originally decided to perform an offline transaction

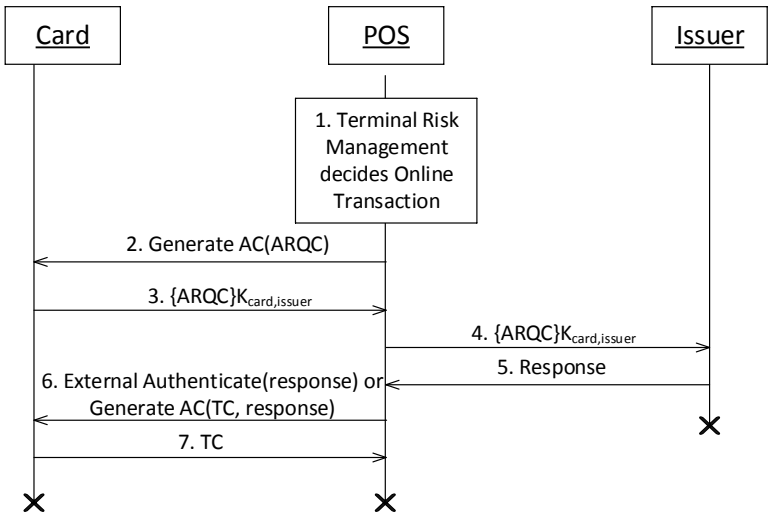


Figure 2.10: Message sequence chart of a successful online transaction where terminal originally decided to perform an online transaction

(message 2). In this case the card cannot decide to perform the transaction offline, but it must send an ARQC (message 4) which can be forwarded by the terminal to the issuer (message 5). Upon receiving approval from the issuer (messages 6 and 7), the card will send the TC to the terminal (message 8). If the terminal cannot forward the ARQC to the issuer (e.g. because of connectivity issues), the terminal can either decide to reject the transaction (because it is not allowed to do offline transactions, or the terminal already knows the card will not allow offline transactions), or try to carry it out offline by sending a TC request.

CDA can be requested by the terminal if it requests a TC or an ARQC. In this case the card returns a TC or an ARQC with SDAD to prove that it has access to the private key of the card, which effectively authenticates the card.

For all cases at all moments, the card can abort the transaction by sending an AAC. The terminal can always abort the transaction by requesting an AAC. The card will always respond with an AAC if this is requested.

### 2.3.6 Cryptogram Generation

The AC generated by the card is static data from the card and transaction data (amount, UN, etc.) from the POS terminal encrypted with an encryption key that is different for each transaction. We call this key the secret transaction key. This transaction key is a symmetric key derived from the ATC and the secret master key $K_{IC}$ on the card. Because the ATC is incremented every transaction, the transaction key is different for each transaction.

# Chapter 3

# EMV Contactless Specifications

This chapter provides a description of the EMV Contactless protocol. Section 3.1 describes the different documents that describe the EMV Contactless specifications and which we used for this chapter. Section 3.2 describes the contactless interface of EMV Contactless cards. Section 3.3 gives a high level overview of EMV Contactless transactions. Section 3.4 gives a complete description of the MasterCard PayPass specifications. Finally, Section 3.5 gives a complete description of the Visa payWave specifications.

## 3.1 EMV Contactless Specification Documents

The EMV Contactless specification is described by the EMV Contactless books [13–15], kernel specifications [16–22], additional documents published by Maestro [34] and to some extent the EMV Contact books [9–12]. Table 3.1 gives an overview of all the EMV books and their number of pages. The EMV Contactless books Book A [13], Book B [14] and Book D [15] are hereafter known as the 'general contactless specifications'. In addition, for each of the six members[9] of EMVco (MasterCard[10] [17], Visa[11] [18], American Express[12] [19], JCB[13] [20], Discover[14] [21], UnionPay[15] [22]) there is a different book describing the corresponding kernel. In addition, there is a seventh book [16] that describes a kernel for a certain subset of Visa and JCB cards. These kernel books vary greatly in size, as shown in Table 3.1. These large differences in size for books on the kernels indicate that the kernels differ significantly in functionality and options.

[9] http://www.emvco.com/about_emvco.aspx?id=156
[10] http://www.mastercard.com/
[11] http://www.visa.com/
[12] http://www.americanexpress.com/
[13] http://www.jcbank.com/
[14] http://www.discovernetwork.com/
[15] http://en.unionpay.com/

| Book | Title | # pages |
|---|---|---|
| EMV Contact | | |
| Book 1 [9] | ICC to Terminal Interface Req. | 189 |
| Book 2 [10] | Security and Key Management | 174 |
| Book 3 [11] | Application Specification | 230 |
| Book 4 [12] | Cardholder, Attendant, and Acquirer Interface Req. | 154 |
| EMV Contactless general | | |
| Book A [13] | Architecture and General Req. | 114 |
| Book B [14] | Entry Point Specification | 47 |
| Book D [15] | Communication Protocol Spec. | 247 |
| EMV Contactless kernels | | |
| Book C-1 [16] | Visa and JCB kernel | 34 |
| Book C-2 [17] | MasterCard kernel | 546 |
| Book C-3 [18] | Visa Kernel | 189 |
| Book C-4 [19] | American Express kernel | 155 |
| Book C-5 [20] | JCB kernel | 128 |
| Book C-6 [21] | Discover kernel | 105 |
| Book C-7 [22] | UnionPay kernel | 80 |

Table 3.1: Overview of all EMV Contact and Contactless books

In addition to the general contactless specifications, there are multiple references to the original EMV Contact specification books. This significant number of total pages used to describe the EMV Contactless specifications forms a significant challenge for describing the payment process in a compact matter.

Because of the significant number of pages used to describe the total contactless payment system, it is not possible to give an exhaustive overview of the system with all seven kernels. Furthermore, the contactless system offers many options and parameterizations of which the majority does not add or break any security features. While the EMV Contact specifications are well studied and extensively described in multiple papers, there is little work done on EMV Contactless. An exhaustive overview was not yet available. For the overview of EMV Contact in Chapter 2 we could use the available overviews from different studies and papers. For the overview of EMV Contactless that we present in the remainder of this chapter, however, we not only used the specification books by EMV, we also extensively tested the Dutch cards we had available, to see and learn how the cards respond in general to the commands described in the specification books. We decoded the responses with the specification books and tried to perform transactions to see how that works. With performing transactions

we verified whether we correctly understood the specification books as they are more than once unambiguous.

MasterCard is the dominant standard in the Netherlands as the Dutch banks issuing contactless cards all use the MasterCard specifications. Therefore, we have placed the focus of this research on the most relevant kernel for the Netherlands (MasterCard Kernel 2) rather than on EMV Contactless in general. In addition, we also describe the Visa kernels (Kernel 1 and Kernel 3) as Vodafone is also issuing contactless Visa cards in the Netherlands.

## 3.2 Contactless Interface

NFC is a collection of standards and technical specifications that defines how two nearby supporting devices can communicate via inductive coupling of radio frequency fields [36]. With a maximum speed of 424 kbps, NFC is much slower than for example Bluetooth or wifi, but NFC needs considerably less power to operate. The range is estimated at proximately 20 cm in theory but in practice a range of 4 cm is more common [37]. Communication requires an initiating device and a transponding device. Both devices can send but only one device can send at a time (also known as half-duplex).

RFID is a technique used to read and write contactlessly from a reader to a RFID tag. NFC specifications are compatible with RFID, so NFC readers can communicate with RFID tags. Newer bank cards contain a RFID tag in order to perform contactless transactions.

A privacy issue could arise because people can be tracked by these contactless cards and possibly linked to their identity without even making physical contact with the card. Bank account numbers should remain private as the Dutch government warns their civilians that disclosure of their bank account number could lead to identity theft and an empty bank account[16]. However, the range of RFID is very limited, so it is unlikely that people will actually be tracking persons successfully without big antennas while the gain of doing this is rather limited.

## 3.3 EMV Contactless Transactions

EMV Contactless transactions are in many ways similar to EMV Contact transactions. One of the major differences is that with contactless cards, some of the transaction steps can take place *after* the card has left the proximity of the reader. The advantage is that this decreases the time needed for the customer to present the card to the reader.

[16] `http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/brochures/2014/10/23/3-stappen-tegen-identiteitsfraude/stappenplan-identiteitsfraude.pdf`

An EMV Contactless transaction begins with the selection of a combination of an application on the card and a kernel on the terminal. An application on the card is a software program that supports the commands, data items and communication protocol for cards as defined by its corresponding kernel in one of the seven kernel books. A kernel on the terminal is a software program that supports the commands, data items and communication protocol for terminals as defined by one of the seven kernel books. Cards can have multiple applications and terminals can have support for multiple kernels. Typically a terminal has support for more than one kernel for compatibility reasons (e.g. in Europe it is very common to support at least Visa and MasterCard), while a card typically has one or two applications.

As certain applications are compatible with certain kernels, a compatible combination has to be selected. This procedure is described in the following subsection.

Figure 3.2 presents an overview of the possible methods to perform a transaction for Visa and MasterCard cards. The first decision is based on the interface through which the card communicates with the terminal (either magnetic stripe, contact chip or contactless). For the magnetic stripe, only one method to perform transactions is possible, which is not further discussed in this thesis. When the card communicates with the contact chip, also only one method for transactions is possible. This is called 'EMV Contact' and is extensively described in Section 2.3. When the contactless interface is used, a decision is made based on the brand of the card. In our overview we only included MasterCard and Visa.

- For Visa cards, one of two kernels is chosen based on the priority setting in the card. If Kernel 1 has the highest priority, a 'Kernel 1 transaction' is performed. There are no Kernel 1 implementations in the Netherlands and therefore not discussed in this thesis. Otherwise, if Kernel 3 has the highest priority and both card and terminal support Contactless Mag-Stripe Mode then a Contactless Mag-Stripe Mode transaction is performed as described in Section 3.5.2. Otherwise, if Kernel 3 has the highest priority and Contactless Mag-Stripe Mode is not supported by either the card or the terminal (or both), a qVSDC mode transaction is performed as described in Section 3.5.1.

- For MasterCard cards, only Kernel 2 can be used. Kernel 2, however, supports two different contactless transaction methods, namely 'Contactless Mag-Stripe Mode' and 'Contactless EMV Mode', as described in Sections 3.4.2 and 3.4.3 respectively. If both card and terminal support Contactless EMV Mode, a Contactless EMV Mode transaction is performed. Otherwise, if both card and terminal support contactless Contactless Mag-Stripe Mode, a Contactless Mag-Stripe Mode transaction is performed. If card and terminal do not share a supported mode, the next application or

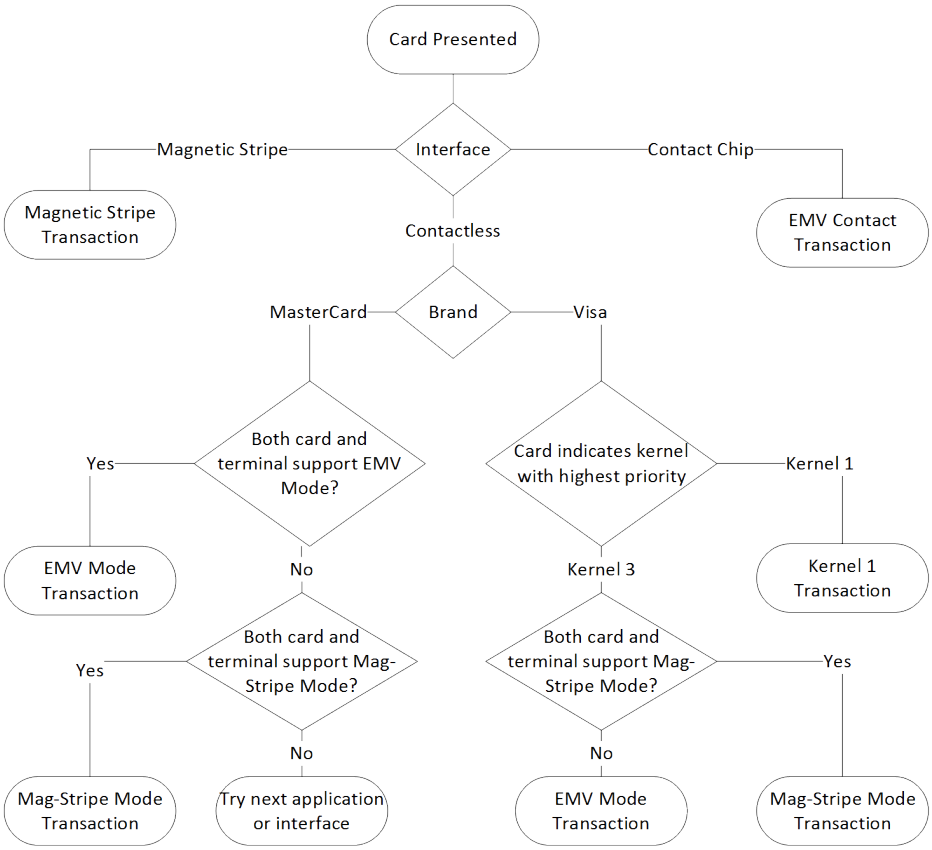


Figure 3.2: Overview of the possible transaction methods for MasterCard and Visa cards including decision making

next interface should be tried depending on the availability of next applications and interfaces.

One of the most interesting differences between the selection of modes for Visa and MasterCard cards is whether Contactless Mag-Stripe Mode is preferred over EMV Mode. Visa cards cannot indicate whether EMV Mode is supported and MasterCard cards cannot indicate whether Contactless Mag-Stripe Mode is supported. For Visa cards, Mag-Stripe is indeed preferred over EMV Mode. Requirement 5.2.2.3 from the Visa specifications [18] states: "*If the reader is both mag-stripe mode-enabled and EMV Mode-enabled [...] and if the card indicates mag-stripe mode is supported [...] then the kernel shall proceed with Mag-Stripe Mode*". Section 3.4.1 from the MasterCard specifications [17] states: "*Based on the response from the Card, in particular the Application Interchange Profile, the Kernel continues with either a Mag-Stripe Mode or an EMV Mode transaction*". From this description it is not immediately clear how the kernel should continue based on different AIPs. Book C-2 [17, Section 6.5] shows that the AIP can

| | | Visa | MasterCard |
|---|---|---|---|
| Contact | Market name | V PAY | - |
| Contactless | Market name | payWave | PayPass |
| | EMV Mode | payWave EMV Mode | PayPass M/Chip |
| | Mag-Stripe Mode | payWave Mag-Stripe | PayPass Mag-Stripe |

Table 3.3: Market names for different systems per brand

indicate EMV Mode. It is not immediately clear how the AIP can indicate this, however, [17, Annex A.1.16] indicates that when bit 8 of byte 2 of AIP is set to 1, then Contactless EMV Mode is supported. This is inconsistent with Book C-3 [18, Annex A.2] which states that bit 8 of byte 2 means that Contactless Mag-Stripe Mode is supported.

Now that we have established how the different transaction modes are selected, we explain these transaction modes in more detail in Sections 3.4 and 3.5.

**Brands and Terminology** Both Visa and MasterCard have EMV Mode and Mag-Stripe Mode available for contactless transactions. Contactless transactions can thus be either EMV Mode or Mag-Stripe Mode transcations. Both EMV Mode and Mag-Stripe Mode can only be performed contactlessly. Contact transactions are always called EMV Contact transactions.

Visa markets their contactless transaction system as 'payWave'. This, however, is not mentioned in the Visa specifications. They do not have special or promo names for the two methods that payWave can use to perform transactions, namely Contactless EMV Mode and Contactless Mag-Stripe Mode. Visa's implementation of EMV Contact is called V PAY.

MasterCard markets their contactless transaction system as 'PayPass'. This name, is also not mentioned in the MasterCard specifications. MasterCard does have a special name for their implementation of Contactless EMV Mode, namely 'PayPass M/Chip'. Their implementation of Contactless Mag-Stripe Mode is simply called PayPass Mag-Stripe. An overview of the names is given in Table 3.3

In addition, MasterCard brands their cards made for certain markets (e.g. Europe) as Maestro and not as MasterCard. Maestro cards can have different requirements, options or restrictions compared to MasterCard branded cards [34].

## 3.4 MasterCard PayPass Specifications

The MasterCard contactless implementation PayPass is extensively described in Book C-2 [17]. Mastercard PayPass cards can be branded 'MasterCard' or 'Maestro', depending on the target market. One of the most important features of

PayPass is that it supports two types of transactions: Contactless EMV Mode transactions and Contactless Mag-Stripe Mode transactions. Contactless Mag-Stripe Mode operates completely different from Contactless EMV Mode or EMV Contact and performs payments based on magnetic stripe-like data obtained from the card. Contactless EMV Mode operates similar to the EMV Contact specifications. For all valid implementations of the kernel, Mag-Stripe Mode is mandatory and EMV Mode is optional [17, Table 3.3]. However, there is an 'only EMV Mode transactions supported' option which disables the Contactless Mag-Stripe Mode [17, Table 3.4]. The PayPass M/Chip requirements [34] offer more clarity: MasterCard branded cards *must* support Mag-Stripe Mode and *may* support Contactless EMV Mode, Maestro branded cards *must* support Contactless EMV Mode and *must not* support Contactless Mag-Stripe Mode. However, the card can indicate whether it supports Contactless EMV Mode in addition to Contactless Mag-Stripe Mode, but it cannot indicate whether it supports Contactless Mag-Stripe Mode [17, Annex A.1.16]. A Maestro branded card must not support Contactless Mag-Stripe Mode but there is no option to indicate this to the reader. This ambiguity in books and published documents seems to indicate that MasterCard initially considered Contactless Mag-Stripe Mode as the default operating mode, later found out this was not a smart idea and disabled it for some cards (at least all Maestro branded cards) but did not update the kernel specifications yet to the new situation.

### 3.4.1 MasterCard Contactless Transaction Initialization

A contactless transaction starts with the terminal selecting an application first and then a mode. This process is described in this section and the message sequence chart is shown in Figure 3.4.

Every contactless transaction begins with the card sending its Answer To Reset (ATR) (message 1). The terminal then selects the Proximity Payment System Environment (PPSE) (message 2). The card responds with its list consisting of an AID with a priority indicator per supported payment application (message 3). The terminal selects the supported application with the highest priority (message 4) and the card responds whether the AID was selected correctly (message 5). The terminal requests the processing options (message 6) and the card responds with the AIP and AFL (message 7). The terminal requests the record indicated by the AFL (message 8) and the card returns these records (message 9).

The AIP on the card can only indicate whether Contactless EMV Mode is supported, not whether Contactless Mag-Stripe Mode is supported. If both the AIP and the terminal indicate that Contactless EMV Mode is supported, the transaction is performed with Contactless EMV Mode. Otherwise, if the terminal indicates that only Contactless Mag-Stripe Mode is supported, the transaction is performed with Contactless Mag-Stripe Mode. If the terminal is set to support only Contactless EMV Mode and the card does not support Contactless EMV

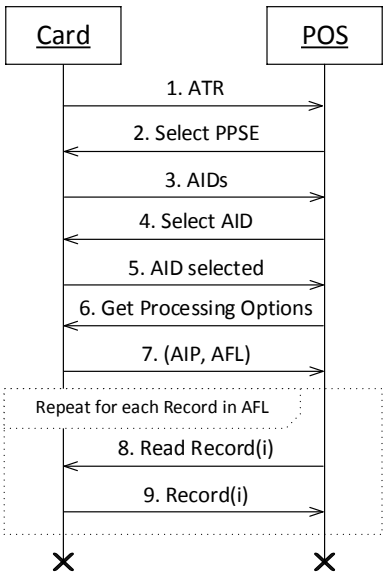


Figure 3.4: Message sequence chart of the beginning of a MasterCard contactless transaction

Mode, the next AID must be selected (message 4). If there is no more next AID to try, another interface (e.g., contact chip, magnetic stripe) should be tried.

### 3.4.2 MasterCard Contactless Mag-Stripe Mode

Mag-Stripe Mode is mainly used for backward compatibility reasons. Existing mag-stripe terminals can be easily extended with an NFC reader to enable contactless transactions. Mag-Stripe Mode does not, however, as opposed to EMV Contact and EMV Mode, authenticate static data on the card. Cardholder verification without a cardholder device or authentication of the transaction data are not possible. Roland and Langer showed a method to successfully clone cards due to problems with the UNs [40].

A message sequence chart of a successful Mag-Stripe Mode transaction is shown in Figure 3.5. If both the AIP and the terminal indicate that mobile CVM is supported and the terminal finds it necessary to perform cardholder verification, the POS Cardholder Interaction Information (PCII) is requested (message 1). The cardholder device sends the PCII to the terminal, which contains information about whether the PIN was entered correctly and possible conflicts (message 2). The terminal requests a cryptographic checksum over an UN with the `COMPUTE CRYPTOGRAPHIC CHECKSUM(UN)` command (message 3). The card responds with the current ATC and a Card Verification Code (CVC), constructed from a secret symmetric key contained in the card (and known by issuer), the ATC and the

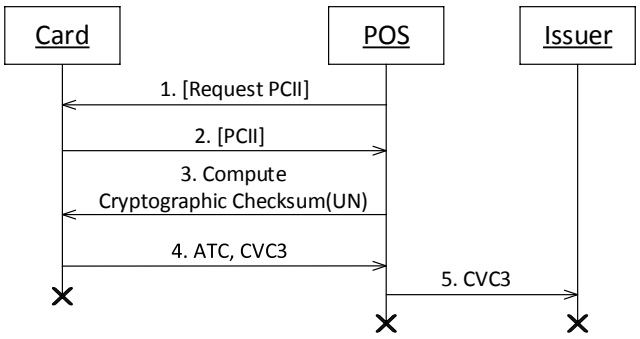


Figure 3.5: Message sequence chart of a successful MasterCard Contactless Mag-Stripe Mode transaction

UN received from the terminal (message 4). The CVC is a dynamic CVC, called CVC3 for contactless transactions, belonging to Contactless Mag-Stripe Mode transactions. The terminal sends the CVC3 to the issuer for verification (message 5). The issuer verifies the checksums and approves the transaction. The CVC3 is to a large extent backwards compatible with the traditional static CVC on the back of the card. The transaction is now complete.

### 3.4.3 MasterCard Contactless EMV Mode

**Card Authentication** When all the necessary files have been read during the initialization described in Section 3.4.1, the terminal risk management determines whether card authentication should take place. SDA is performed in the same way as described in Section 2.3.3 and CDA is performed together with the transaction. This means that in both cases no additional interaction between card and terminal is necessary. As mentioned earlier, DDA is no longer supported for contactless transactions.

**Cardholder Verification** The terminal risk management also decides which CVM should be performed. If both card and terminal indicate that On-Device Cardholder Verification is supported and the terminal decides it is necessary to perform cardholder verification, the transaction is performed with On-Device Cardholder Verification (On-Device Cardholder Verification thus has priority over the CVM list). The terminal passes the argument 'offline (plaintext) PIN was performed' with the `GENERATE AC` command. The cardholder device then knows On-Device Cardholder Verification must be performed and will perform the verification. The cardholder device only responds with a TC or ARQC to the terminal if the cardholder is authenticated on the device. It responds with an AAC if the

verification is not performed correctly. It looks like this method could be exploited for offline transactions. A Man-In-The-Middle (MITM) could potentially manipulate the messages to trick the card into thinking the terminal did not request On-Device Cardholder Verification and it can trick the terminal into thinking the On-Device Cardholder Verification was performed correctly. However, the issuer will notice this when it receives the encrypted message from the card.

If the terminal decides cardholder verification is necessary but On-Device Cardholder Verification is not supported by either card or terminal or both, either online PIN verification or signature verification must be performed, depending on the capabilities of the terminal and the priorities indicated by the card. When online PIN is chosen as CVM, this is indicated together with the `GENERATE AC` command without actually being performed at this point. When signature verification is chosen as CVM, this is also indicated together with the `GENERATE AC` command. The printed receipt then should contain a signature line and this should be signed by the cardholder after the transaction is completed.

**Completing the Transaction** The terminal risk management also determines whether the transaction should be performed online or offline. All these decisions (card authentication method, cardholder verification and online or offline transaction) are sent with at least the amount, currency, country and date of the transaction together with the `GENERATE AC` command. The card responds with an AC. There still are three possible ACs just as with EMV Contact, although they have a slightly different meaning. A TC is proof that the transaction took place and indicates that it was performed offline. This is different from a TC with EMV Contact where it could also be proof that an online transaction took place after the card received online authorization from the issuer. An ARQC indicates that the card or the terminal finds it necessary that the issuer authorizes the transaction online. For a successful transaction, an ARQC is not followed by a TC in contrast to EMV Contact transactions. An AAC indicates that the transaction was aborted.

When the terminal requests a TC (message 2 in Figure 2.8) and the card risk management finds this acceptable (action 3 in Figure 2.8), the card responds with a TC (message 4 in Figure 2.8). The transaction is now completed offline and the interaction is terminated. However, the card responds with an ARQC (message 4 in Figure 3.6) if the card risk management determines the transaction should be performed online (action 3 in Figure 3.6). The interaction is now also terminated but the transaction still needs to be completed online. The terminal forwards the ARQC with optionally the PIN to the issuer (message 5 in Figure 3.6) which responds with an 'approved' or 'declined' message (message 6 in Figure 3.6).

The terminal can also decide the transaction needs to be performed online by requesting an ARQC (message 2 in Figure3.6). In this case the card cannot respond with a TC but must return an ARQC for a successful transaction (mes-

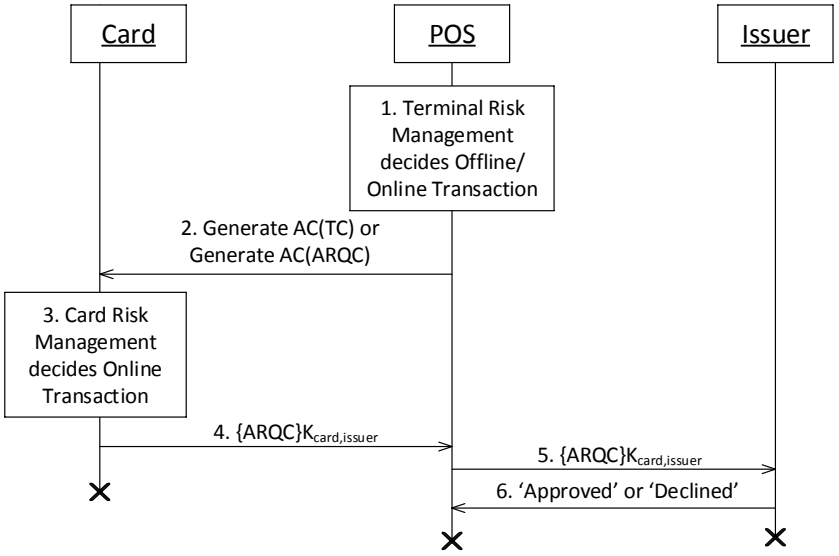


Figure 3.6: Message sequence chart of a contactless transaction where the card forces an online transaction

sage 4 in Figure 3.6). The interaction between card and terminal is terminated and the terminal forwards the ARQC to the issuer optionally with the PIN. The issuer responds with an 'approved' or 'declined' message (message 6 in Figure 3.6)

The PIN is optionally included with the ARQC to the issuer depending on whether the terminal indicated to the card the PIN should be verified online. If the terminal indicated this to the card, this decision is included in the returned ARQC so that the issuer learns that online PIN should be performed. The terminal thus must include the encrypted PIN together with the ARQC when it indicated to the card that the PIN should be verified online, otherwise it should not include the PIN.

Just as with EMV Contact, both the card and the terminal can at any moment terminate the transaction by respectively returning an AAC or by requesting an AAC.

**Comparing EMV Mode with EMV Contact** We first give an overview of the (significant) differences between EMV Mode and EMV Contact. After this overview each difference is further discussed.

- Card authentication methods SDA and CDA are still supported, however, DDA is no longer supported,

- Cardholder verification through offline PIN verification is no longer considered suitable,
- Cardholder verification with a cardholder's device is supported for EMV Mode transactions,
- The second `GENERATE AC`, needed for traditional EMV Contact transactions, is no longer supported,
- EMV Mode introduces Data Storage, with which terminals can write and read data to the card,
- Torn transactions can be recovered with Contactless EMV Mode (transactions are called 'torn' when the card is removed before the communication was done),
- EMV Mode offers functionality for offline card balance reading,
- New limits are introduced for the reader to determine whether transactions can be performed contactlessly, with or without cardholder verification and whether the transaction must be performed offline or online.

Older cards may support SDA but new cards must only support CDA [34]. DDA is not supported by PayPass, probably because with DDA additional messages are needed to be sent increasing the interaction time between terminal and card. In particular, SDA and CDA do not need additional messages to authenticate the card.

Regarding cardholder verification, Section 5.9.3 of Book A [13] states that "online PIN and signature may be supported − offline PIN is not suitable due to the card in field timing issues." However, Appendix A.1.35, A.1.36 and A.1.150 in Book C-2 [17] describe how the kernel can support offline PIN verification (both plaintext and encrypted). So although it is mentioned offline PIN is not suitable, it can still be supported. Furthermore, it is also still possible to do no CVM.

A new cardholder verification method is introduced and is called 'On-Device Cardholder Verification'. It uses a mobile device from the cardholder to verify the identity of the cardholder. The device simply indicates to the terminal that On-Device Cardholder Verification was performed. How the device should verify the identity is not specified, although it is mentioned in the MasterCard Best Practices [35] that the PIN should not be entered on a keypad or touch screen of a mobile device and that "*it is not yet known if mobile device keypads will ever be appropriate for PIN capture*".

To decrease the time needed for the interaction between card and reader, the transaction process is shortened. As a result, only one AC is requested by and sent to the terminal. When the terminal requests an online transaction or when

the card decides the transaction needs to be performed online (by requesting or sending an ARQC), no additional TC needs to be requested or sent. Instead, the ARQC is sent to the terminal and that finalizes the transaction. As a result, as soon as the card sends the cryptogram it can be removed from the proximity of the reader.

EMV Mode offers Data Storage, a new feature that allows the terminal to use a 'scratch pad' on the card. This features is available in two types. The first one, Standalone Data Storage, uses dedicated commands, `GET DATA` and `PUT DATA`, to read and write to the scratch pad. The second type, Integrated Data Storage, does not use extra commands, but it is integrated in the existing commands `GET PROCESSING OPTIONS` and `GENERATE AC`. This potentially saves time as these commands are already needed for the transaction anyway. Data Storage potentially makes the system more vulnerable as the attack surface is increased. At the very least it implies some potential privacy issues, as readers can write anything on the card. This might include locations, performed transactions and customer numbers. In addition, it might also be possible to perform a buffer overflow attack on the card, which might have more serious consequences.

The card optionally has a log with torn transactions which the terminal can read. For this, a new command, `RECOVER AC`, is introduced. When a tear occurs after the card has received the `GENERATE AC` command and before the terminal has received the AC, the transaction can be recovered. This way the transaction does not have to be completely restarted. This might save time and is probably implemented because RFID communication might be less reliable than communication with a contact chip.

The card optionally stores the balance of the account, so that the balance can be printed on a receipt or displayed on a screen. This only seems useful when transactions are performed offline, because with online transactions the issuer could indicate the current balance. Furthermore, this feature seems to open the door for privacy breaches while it adds little to none functionality. Debit accounts potentially store a large amount of money and criminals can learn this amount contactlessly and anonymously when they are in the proximity of the card. This feature could help criminals by choosing targets with large amounts of money on their accounts.

When one of the two supported traditional CVMs is used (online PIN or signature), the actual verification is completed after the interaction with the card is complete. This already was the case with the signature verification of course since the signature is supposed to be written on the printed receipt. However, On-Device Cardholder Verification "*is completed before the interaction begins*", according to PayPass M/Chip Requirements document [34], which is counterintuitive because before the interaction begins, the amount is not known, effectively making the verification significantly less useful.

**Limits** An EMV Mode supporting terminal has the following transaction limits defined which are not available for EMV Contact:

- *Reader Contactless Floor Limit*, indicates the transaction amount above which transactions must be authorized online,
- *Reader Contactless Transaction Limit*, indicates the transaction amount above which the transaction is not allowed,
- *Reader Contactless Transaction Limit (No On-device CVM)*, indicates the transaction amount above which the transaction is not allowed, when on device cardholder verification is not supported,
- *Reader Contactless Transaction Limit (On-device CVM)*, indicates the transaction amount above which the transaction is not allowed, when on device cardholder verification is supported,
- *Reader CVM Required Limit*, indicates the transaction amount above which CVM must be performed.

These limits are used for terminal risk management to let the terminal decide whether CVM should be performed and whether the terminal should go online.

## 3.5 Visa payWave Specifications

The Visa payWave specifications are described in detail in Book C-3 [18]. Visa payWave transactions start with the selection of the AID by the POS terminal. The card responds whether the selection of the application was successful. Included in this response is the card's Processing Options Data Object List (PDOL). The PDOL consists of the data objects that the card expects from the POS terminal when the POS terminal issues the `GET PROCESSING OPTIONS` command.

Depending on the response of the card and the capabilities of the POS terminal, the POS terminal chooses one of the following four options, which are described in more detail in the remainder of this section:

- The transaction is performed contactless with *qVSDC Mode*,
- The transaction is performed contactless with *Mag-Strip Mode*,
- The transaction is performed contactless with *VSDC Mode*,
- The transaction is performed with the contact chip or magnetic stripe (Contactless not supported).

The four choices are listed in descending priority. If, for example, the card and the POS terminal find both qVSDC mode and Mag-Stripe Mode appropriate, the transaction will be performed with qVSDC Mode.

### 3.5.1 Contactless qVSDC Mode

With qVSDC, the card provides the AC directly with the response to the `GET PROCESSING OPTIONS` (see Figure 3.7). The outcome of the card's risk management, together with the capabilities of the POS terminal, determines whether the transaction is performed online or offline:

- If the transaction is performed online (see Figure 3.7), the AC is an ARQC. The terminal needs to go online to provide the ARQC to the issuer. The issuer then indicates to the POS terminal whether the transaction is accepted or rejected, or whether the PIN is needed.
- If the transaction is performed offline, the AC is a TC. The terminal additionally needs to check the integrity of the TC, as it cannot decrypt the TC itself. Therefore, the card performs fast Dynamic Data Authentication (fDDA), which creates a signature over the transaction details. For fDDA, the card needs to have a private key and asymmetric cryptography support. The public key of the card is signed by the issuer, and this certificate is also available on the card. The signature over the transaction details and the certificate can be sent directly with the response to the `GET PROCESSING OPTIONS`, or otherwise the response contains the AFL. The records included in the AFL then must contain the signature over the transaction details and the certificate with the public key of the card.

### 3.5.2 Contactless Mag-Stripe Mode

PayWave supports a legacy mode called Mag-Stripe Mode. Old POS terminals, based on transaction with the magnetic stripe, do not need to be fully replaced in order to support payWave. The information that is exchanged between card and POS terminal has many similarities with the information shared in a traditional magnetic stripe transaction. However, if the POS terminal has requested an online transaction, then the card not only provides the magnetic stripe-like data, but also generates an AC. Otherwise, if an offline transaction is requested, then the card also provides the magnetic stripe-like data, and in additional it provides a dynamic CVC.

Figure 3.8 shows a complete online Contactless Mag-Stripe Mode transaction. The POS terminal chooses this Contactless Mag-Stripe Mode if the AIP, contained in the response of the card to the `GET PROCESSING OPTIONS` command, indicates that the transaction must be Contactless Mag-Stripe Mode. The AC can be included in the response to the `GET PROCESSING OPTIONS` command, otherwise the response indicates which additional records should be read by the POS terminal. If there is no AC in the response, the POS terminal will read the records mentioned in the response, and the card returns a record that contains the AC.

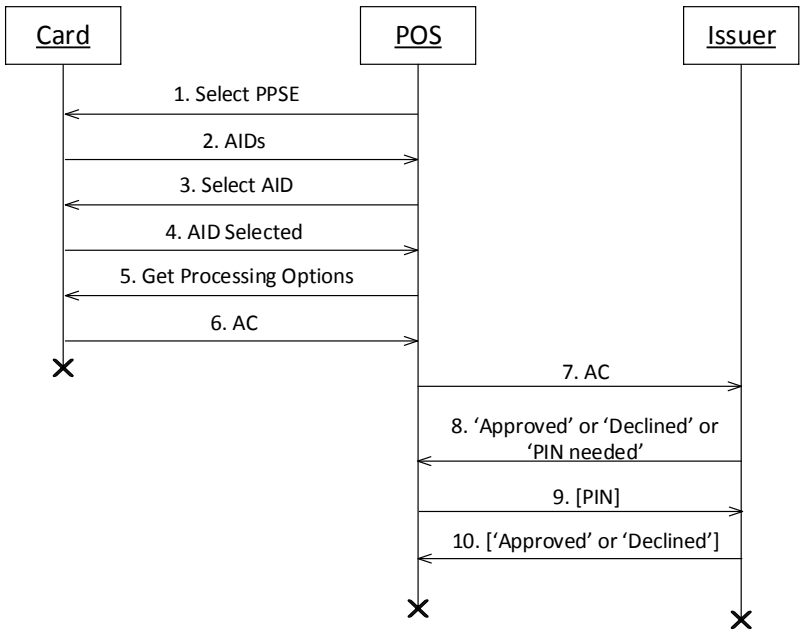


Figure 3.7: Communication sequence of an online qVSDC transaction

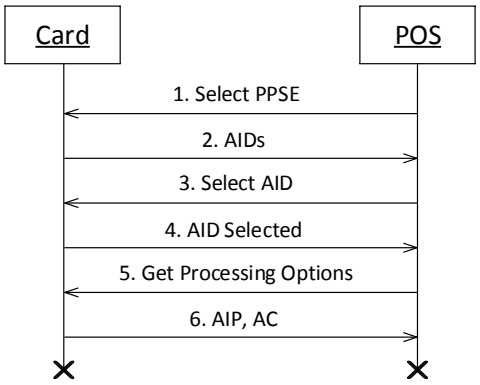


Figure 3.8: A complete online transaction performed according to the payWave Contactless Mag-Stripe Mode specifications

### 3.5.3 Contactless VSDC Mode

Visa offers the possibility to perform a transaction contactlessly exactly according the EMV Contact specifications as described in [9–12], and summarized in Chapter 2.3. The only difference is that they are sent over an NFC-channel instead of

via the contact chip of the card.

### 3.5.4 Contactless Not Supported

If the terminal decides the transaction should be performed with the contact chip or magnetic stripe, then the transaction is not performed according to the payWave specifications. Transactions performed with contact chip are described in Chapter 2.3. Transactions performed with the magnetic stripe are not described in the EMV specifications and are outside the scope of this thesis.

# Chapter 4

# Related Work

This chapter describes all the known attacks on EMV Contact and EMV Contactless. Section 4.1 describes the known attacks on EMV Contact and Section 4.2 describes the known attacks on EMV Contactless.

## 4.1 EMV Contact Attacks

EMV Contact cards are a well-studied topic. Many attacks have been found and documented by researchers. Some attacks have many limits and are not too practical for attackers to exploit, but some attacks are relatively easy to exploit and encountered frequently in real world situations.

Many of the attacks on EMV Contact might apply as well to EMV Contactless. The attacker model found most in the literature for EMV Contact attacks is the Dolev-Yao model. A Dolev-Yao attacker is able to, among others, impersonate identities, eavesdrop, alter, inject, replay and redirect messages [6]. The attacks on EMV Contact that are described in this section indeed use fake or cloned cards, MITM devices, counterfeit or hacked terminals and criminal shop owners.

### 4.1.1 Cloning Cards with Static Data Authentication

The first attack on EMV exploits the static data authentication method in EMV cards [2]. Because the authentication data that is sent from the card to the terminal is static, it is exactly the same every transaction. If a fake card can provide that authentication data to a terminal, the terminal will trust this fake card to be genuine because it has no way to know it is not the original card. Because the terminal trusts the card, it finds it not necessary to verify the provided PIN online, but it trusts the chip on the card to verify it. The response of the card is, however, not authenticated so if the card is indeed a fake, it can be programmed to always respond that the PIN was correct for every possible

entered PIN. When finalizing the transaction, the card sends the TC, which is a message authentication code (MAC) that assures the integrity of the transaction details and is encrypted with the symmetric key shared between bank and card to prove authenticity of the card. A fake card does not know this encryption key, but neither does the terminal, so it has to check with the issuer to verify it. The fake card can simply provide a fake TC as the terminal cannot check this MAC if the transaction is performed offline.

Whenever the terminal goes online to provide the fake TC to the bank, the bank will immediately see that the TC from the fake card does not use the symmetric key shared with the bank and the genuine card, and the transaction will be rejected by the bank. By this time, however, the customer with the fake card has already left the shop with the purchased merchandise and the merchant is left with nothing.

A situation where static information is recorded and replayed at another time or place is known as a replay attack. These fake cards that accept all PINs are also known as yes-cards.

The only real defense to stop this kind of attack is to simply disable SDA. This is an effective countermeasure as it instantly disables this attack and it is also feasible since its successors DDA and CDA are already supported by the EMV specifications. The biggest downside is that cards need to be able to do asymmetric encryption with DDA and CDA, so older cards need to be replaced. In addition, cards with asymmetric encryption capabilities are more expensive than cards that do not support asymmetric encryption.

**In the Netherlands** SDA is no longer used in the Netherlands, so cloning cards with SDA is not possible in the Netherlands.

### 4.1.2 Faking Transactions with Dynamic Data Authentication

DDA is an improved version of SDA. DDA uses a challenge-response mechanism with asymmetric encryption to ensure freshness of the data to authenticate the card. This disables the possibility to replay the authentication data and to authenticate with a fake card.

The TCs delivered from the card to the terminal, however, are not signed with the private key of the card but again encrypted with the symmetric key shared between card and issuer, unknown to the terminal. So again, if the terminal is an offline terminal, or card and terminal agree that the transaction can be performed offline, any TC will be accepted by the terminal.

The only real defense is to disable DDA. This is a feasible solution as firmware of most terminals can be upgraded to disallow DDA. The alternative to SDA/DDA is CDA, which is also part of the EMV Contact specifications. With CDA, trans-

action parameters are also signed by the private key of the card, and can thus be checked by the terminal for authenticity. One of the biggest disadvantages of CDA is that the transaction time increases because more data has to be signed and verified, but with modern electronics this is not really an issue anymore.

**In the Netherlands** DDA is not used in the Netherlands. Furthermore, offline transactions are not allowed in the Netherlands. Therefore, this attack is not possible in the Netherlands.

### 4.1.3 Eavesdropping

Because the EMV Contact protocol does not provide end-to-end encryption, much critical information can be eavesdropped. One of the first papers describing this serious flaw is [1]. The authors identify the need to trust the terminal, which is not in control of the customer using it, as the main source of weakness about the EMV system. The authors mention four methods to record the data from the card in order to create a cloned card. For example, terminals can be hacked, so that they record data from the smart card including the entered PIN. Furthermore, terminals can be counterfeit, so that they look like legitimate terminals but in reality cannot perform any real transactions, but only record data from the card. The authors also describe a scenario where a customer would hand over his card to a malicious shop owner, who secretly swipes it through a counterfeit reader, which records the data from all cards that it reads, before using the legitimate terminal. Finally, terminal skimmers can be used, which are very small devices that record the communication between card and terminal. These devices can be placed on top of legitimate terminals in a way that it is not suspicious for unknowing users. Furthermore, the authors describe five methods to learn a customer's PIN. Cameras can be installed to record anything the customer enters, both hacked and counterfeit terminals can record all keys pressed, skimming devices can record the PIN in case a transaction is performed with plaintext PIN verification and finally people can simply look over the shoulder of the person who enters his PIN.

**In the Netherlands** Some of the attacks described in [1] have occurred in the Netherlands. For example, counterfeit readers were encountered[17] and skimming occurred on a large scale with skimming devices[18]. Lately, there have been few reports of these kind of attacks, presumably because copying cards is no longer possible and banks have taken measures to prevent counterfeit readers. However, criminals in the Netherlands still watch over the shoulder of victims to learn the PIN[19].

[17] https://www.security.nl/posting/34129/Skimmers+manipuleerden+kaartlezers+in+ABN-Amro+filialen

[18] http://www.politie.nl/onderwerpen/skimming.html

[19] http://www.rtvnoord.nl/artikel/artikel.asp?p=138225

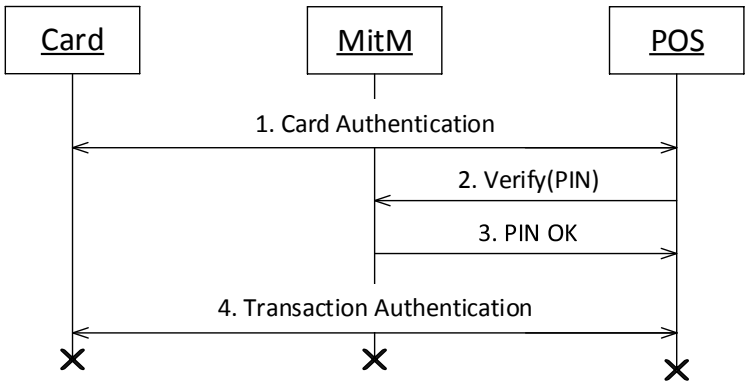


Figure 4.1: Message Sequence Diagram PIN Bypass attack with Man-in-the-Middle (copied from [38])

### 4.1.4 PIN Bypass

In the paper 'Chip and PIN is broken' the authors describe a PIN bypass attack [38] (see Figure 4.1). Because the response of the PIN verification performed *on* the card is not authenticated, it is not only possible to create a yes-card as described in [2], but also to spoof the response of the card. The MITM can trick the card into believing no PIN code is entered while sending a message to the terminal that the PIN was correctly entered (messages 3 and 4). As a result, the PIN try counter is not decremented. With this attack, there is no need to eavesdrop information but any (stolen) card can be used without knowing the PIN. Additionally, because the terminal thinks a PIN was entered, the receipt (if any) will indicate that a PIN was successfully entered.

**In the Netherlands** The Radboud University Nijmegen tried to perform this attack some time ago, and succeeded. However, not very long after that, all POS terminals in the Netherlands have been updated so that the PIN bypass attack was no longer possible. Therefore, it is very unlikely that there still are POS terminals and banks in the Netherlands that are vulnerable to this attack.

### 4.1.5 Relay Attack

The authors of [2] also describe a relay attack. This attack uses the fact that customers have no guarantee that the display of a terminal is actually showing correct information. As a consequence, customers have no guarantee whom they are doing business with and what amounts are transferred. This exploit is considered more sophisticated because it is a real-time attack and the timing is

crucial. The scenario that is described is one where an attacker is waiting in a jewelery store to purchase a very expensive diamond, and a crooked restaurant waiter that uses a counterfeit terminal. The counterfeit terminal in the restaurant forwards the data stream to the attacker with a fake card in the jewelery store. All transaction details from the jewelery store are forwarded to the terminal in the restaurant. The counterfeit terminal does not show the transaction details of the actual transaction (the transaction carried out in the jewelery store) but shows an amount that is compliant with the bill of the meal. Now the guest of the restaurant thinks he is paying for his meal while in reality he is paying for a diamond he is never even going to see. This real-time exploit works better in restaurants than in other stores because in restaurants the terminal operator can determine to a greater extent when the customer is going to enter his PIN.

**In the Netherlands** To the best of our knowledge, there are no reports of relay attacks performed in the Netherlands. However, we do believe they are possible, but finding a crooked restaurant waiter and performing this attack is probably too difficult with respect to the potential gain.

## 4.2 EMV Contactless Attacks

This section describes the known EMV Contactless attacks and to what extent the EMV Contact attacks are applicable to EMV Contactless. Section 4.2.1 discusses how and whether the EMV Contact attacks from Section 4.1 relate to EMV Contactless. Sections 4.2.2, 4.2.3 and 4.2.4 describe the known attacks on EMV Contactles. Furthermore, Sections 4.2.5 and 4.2.6 describe two relay attacks performed on NFC devices.

### 4.2.1 EMV Contact Attacks on EMV Contactless

This section describes whether and how the EMV Contact attacks from Sections 4.1 relate to EMV Contactless.

**Cloning Cards with SDA** SDA is still supported by kernels 3, 4, 5 and 7 [18–20, 22]. However, the dangers of SDA are widely known and it is unlikely that banks will issue contactless cards that accept SDA.

**Faking Transactions with DDA** DDA is not supported by any of the kernels. Therefore, attacks on DDA are not applicable to EMV Contactless.

**Eavesdropping** EMV Contactless still does not support end-to-end encryption, so any information shared between POS terminal and card can be intercepted. Eavesdropping is actually easier with contactless cards as information

is sent contactlessly and eavesdropping devices do not have to make physical contact with the cards.

**PIN Bypass** Offline PIN verification performed on the card is not supported by any of the seven kernels. Therefore, the attack described in [38] is not applicable to EMV Contactless. However, EMV Contactless introduces On-Device Cardholder Verification, which basically is offline PIN verification performed on a mobile device (such as a mobile phone). On-Device Cardholder Verification seems vulnerable to the same general idea, as the verification result is still unauthenticated. However, we could not verify this because we are yet to encounter an implementation that uses the On-Device Cardholder Verification.

**Relay Attack** The EMV Contactless specification does not introduce countermeasures against relay attacks. Therefore, relay attacks as described in [2] are still possible. Chapter 6 describes our relay attack on EMV Contactless.

### 4.2.2 Pre-play & Downgrade Attack on Mag-Stripe Mode

Roland and Langer [40] describe a method on how to create functional clones of a card including the necessary data to perform Contactless Mag-Stripe Mode transactions. The only dynamic information necessary to perform a Contactless Mag-Stripe Mode transaction is the dynamic CVC. The rest of the necessary information is static and can simply be eavesdropped from a legitimate transaction or can be requested by any contactless card reader as no authentication is required. The CVC is requested by the terminal by issuing a `COMPUTE CRYPTHOGRAPHIC CHECKSUM` command. It is computed from the UN provided by the terminal, the ATC known by the card and a secret symmetric key shared by card and issuer.

The UN has a maximum of four bytes. So in theory there are $2^{32}$ UNs possible. However, because these bytes are binary coded decimals, the number of possibilities is roughly 43 times as small as previously mentioned. In addition, most cards have very limited space allocated for the UN together with the ATC. Because this available space generally is seven digits and the ATC generally is four digits, only three digits remain for the UN. This comes down to a total of only $1,000$ different possibilities for the UN. Indeed the authors tested six cards of which four use three digits, one card uses two digits and one card only uses one digit for the UN.

Because of this small number of possible UNs, an attacker could harvest CVCs computed with all possible UNs if they have access to the card. Of course, with every computation of a CVC, the ATC gets incremented. Once the attacker has a legitimate combination of (UN, ATC, CVC) for all possible UNs, they can load these combinations onto their cloned card. Now the cloned card can be used with real transactions. The card identifies itself by sending the cloned

static data from the original card. The terminal now requests a CVC by issuing the `COMPUTE CRYPTHOGRAPHIC CHECKSUM` command with a terminal generated UN. The cloned card looks the appropriate combination of (UN, ATC, CVC) up (corresponding to the received UN), and sends the ATC and CVC back to the terminal. The transaction is now finalized.

Because PayPass compliant cards and terminals within the Single European Payment Area nowadays must support at least Contactless EMV Mode (with optional support for Contactless Mag-Stripe Mode), and because Contactless EMV Mode is preferred above Contactless Mag-Stripe Mode, this might seem as not an issue. However, because the AIP is not authenticated, the cloned card can claim to the terminal that it does not support Contactless EMV Mode, forcing the terminal to initiate a Contactless Mag-Stripe Mode transaction. This downgrade attack significantly increases the impact of the Pre-play attack as it becomes more widely applicable, even with newer cards.

### 4.2.3 Offline PIN Verify Attack

Emms et al. [7] found some vulnerabilities when offline PIN verification is enabled on contactless cards. In that case, an attacker can try to guess a PIN without the owner ever knowing it. The owner of the card never learns about this attempt because every time the card is used in a POS terminal or Automated Teller Machine (ATM), the retry counter is reset. The attacker, however, does have to perform a check before he issues an offline PIN verification, because if the PIN counter is 1 and another incorrect PIN is tried, the card gets blocked. Luckily for the attacker, the PIN retry counter can also be read contactlessly, so if he performs a simple check, the card cannot get blocked by his attack. A typical PIN retry counter is three, so in general two PINs can be tried each time.

Bonnneau et al. [4] show that with six guesses, an attacker has an 8.23% chance of guessing the PIN correctly. However, we think that percentage is rather irrelevant for the attack described in [7]. Three of the mentioned six PIN tries are not applicable because the card cannot be used in an ATM in the scenario that is described. Furthermore, assuming the attacker does not want to block the vast majority of the cards he encounters, he can only try two PINs before the owner uses their card in an ATM or POS terminal. However, we completely agree the offline PIN verify function is redundant for the contactless interface, does increase risk to some extent and therefore should not be available.

### 4.2.4 Harvesting High Value Foreign Currency Transactions

Emms et al. [8] found that some British Visa credit cards do not limit the amount for an offline contactless transaction when a foreign currency is requested. The

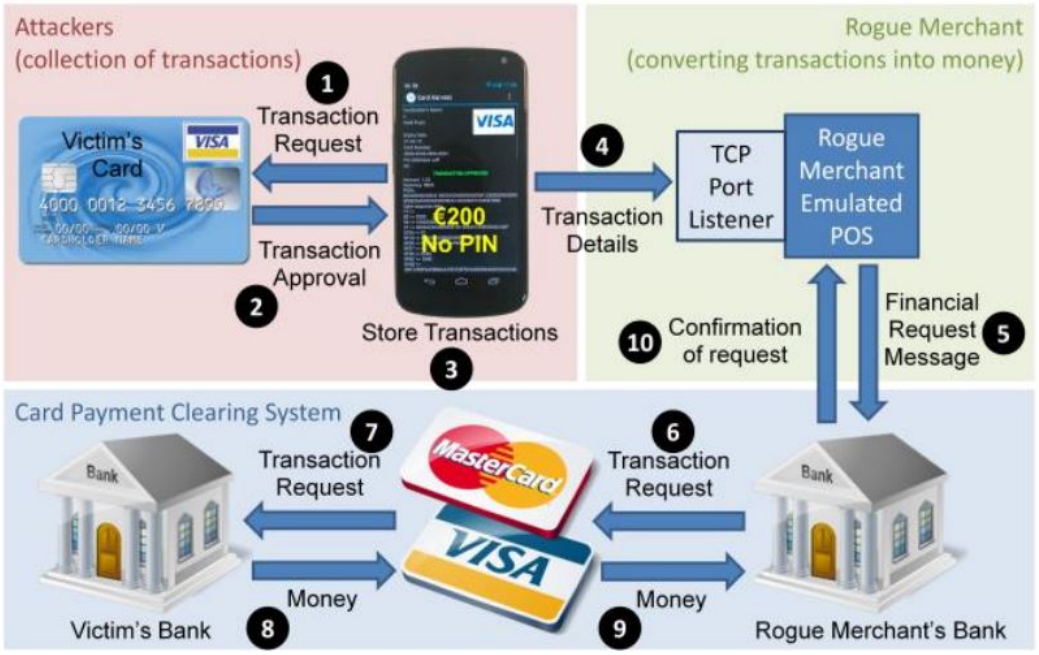


Figure 4.2: Ten steps process to harvest high value TCs (copied from [8])

authors created a ten-steps process for their method, shown in Figure 4.2. First they created an app for NFC-enabled Android phones with which they can perform transactions to harvest Transaction Cryptograms (steps 1 and 2). These TCs are stored (step 3) on the phone and sent later to a rogue merchant with an emulated POS terminal (step 4). This merchant is rogue in the sense that he is registered as a legitimate merchant and thus can make legitimate transactions, but he also has an emulated POS terminal which he can use for false transactions. POS terminals do not send TCs or ARQCs directly to the bank: these are first converted together with some other data (such as date, merchant id, etc.) to a financial presentment. The emulated POS terminal converts the harvested TCs to financial presentments and sends them to its bank (step 5). This bank clears the financial presentment with the bank of the victim and receives the money. Now the rogue merchant's account contains the money from victim.

In theory this could work, but the authors did not try it in practice. We have reasons to believe that it is not possible to actually perform their attack. Although the authors tried steps 1 to 4, we think they did not try or validate the most important and difficult part of their attack. Acquiring TCs from bank cards, sending them to an Internet server and converting them to financial presentments is not much of a challenge: the process of performing transactions with an NFC-enabled smart phone is rather straightforward (as shown by [40]) and converting TCs to financial presentments is extensively described in [41] and ISO 8583-

1:2003 [24]. We have identified the following requirements for their attack as the real, difficult, challenges, on which we elaborate more later in this chapter:

- The emulated POS terminal needs to be authenticated to the bank (step 5 in Figure 4.2),
- The bank must accept high value offline transactions where no PIN was entered (step 7),
- To achieve the claimed 'large scale' the attackers need to solve the ATC inconsistencies (step 1 and 2),
- Attackers need to find money mules not only for opening bank accounts, but also for registering merchant accounts.

**Authentication Challenge** POS terminals and banks use secure channels for communication and they authenticate themselves to each other. This makes it difficult to create a functional rogue POS terminal. Indeed, to the best of our knowledge, there are no reports of a successful rogue POS terminal that could convert TCs into real money.

**TC Acceptance Challenge** Even if the attackers succeed in sending their acquired TCs to the bank, we seriously doubt if these TCs can be cleared. We have multiple reasons to question this method.

- There are hard limits defined for the maximum amount of contactless transactions. Transactions without PIN are only possible up to €25 or equivalent in foreign currency[20,21].
- POS terminals contain 'Reader CVM Required Limits' so that large amounts cannot be paid without a PIN code. These POS terminals are certified and are thus 'guaranteed' to have these limits set to certain amounts. If an emulated POS terminal uses a different limit, the banks will know.
- Equens, the European largest payment processor[22], does not accept any contactless transaction with a value above €25 without a PIN code[23].
- The authors claim their cards are able to make transactions up to £85. These transactions, however, cannot be cleared as there is a hard limit in the UK at £20. This limit was initially set at £10, later increased to

[20] http://www.mastercard.com/us/merchant/pdf/TPR-Entire_Manual_public.pdf
[21] https://www.paypass.com/pdf/public_documents/maestro%20pp%20imp%20req%20june%2007.pdf
[22] http://www.equens.com/
[23] http://www.equens.com/cards/nextgenerationservices/contactless.jsp

£15 and finally set at £20. The reports in the media[24,25] suggest these limits are adjusted in the back end of the system and in the POS terminal, strengthening our claim that the back end system does check the limit of a contactless transaction.

- In some countries it is not possible to perform offline transactions (e.g. The Netherlands).

**ATC Inconsistencies** The authors claim this can be used on a very large scale, as transactions can be harvested, and sent to the bank later. The TCs, however, contain the date of the transaction (in this case given by the phone of the attacker). The attackers could of course choose a date in the future on which they wish to clear all their TCs, and feed this date to all bank cards. However, this date cannot be too far in the future, because when the bank card is used for a legitimate transaction after it was harvested, the bank knows the real ATC at that given moment. We give an example to clarify this more:

The attacker chooses to clear their TCs on the tenth of September. He performs the first step of his attack on the first of September, but of course 'tells' the card it is the tenth of September when he is harvesting the TCs. The card uses its real ATC, let's say it is 100 at the time of this transaction. So the attacker has a TC with ATC at 100 and the date at the tenth of September. Now the victim uses the card on the fifth of September for a legitimate transaction. A transaction is sent to the bank with the date set at the fifth of September and the ATC at 101. Now at the tenth of September, the bank receives the transaction from the attacker, with the date of the tenth of September but with an ATC that is lower than it was on the fifth of September. The bank knows this is not possible and rejects the transaction.

**Money Mules** Furthermore, the attacker needs to find money mules for laundering the money. Finding money mules in foreign countries might be easier than finding them in the UK, but they still need to register as a merchant in order to clear TCs with the banks. This increases the difficulty as identification is required for this step. If an attacker is going to commit identity theft, we doubt this is the most profitable or easy way to make money.

We question the claim of the authors that they discovered a new vulnerability of the EMV protocol. We think this is not a vulnerability of the protocol, as the protocol does not even define limits of transactions, it only defines a method to set a limit. How this limit is set, is up to the card issuers. We do believe their

[24] http://www.theukcardsassociation.org.uk/wm_documents/Contactless%20limit%20-%20final.pdf

[25] http://www.newsroom.barclays.com/Press-releases/Contactless-limit-increases-6ae.aspx

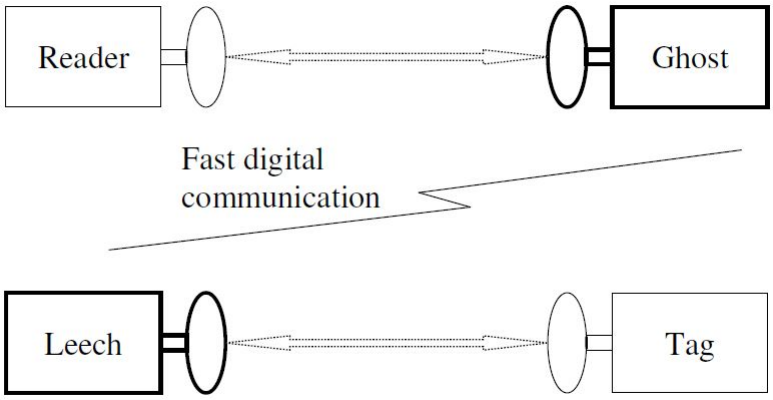


Figure 4.3: Relay setup used by Kfir and Wool (copied from [32])

finding is a sloppiness of the UK cards, as offline limits should have been set for all currencies, and not only for Pounds.

The authors disregard the risk management that takes place at the bank's site and we think it is very likely the bank will not clear the harvested TCs. We think the authors could easily verify their attack without doing anything illegal, with a befriended merchant and their own cards. It would even be very interesting to learn whether they can actually find a place where they can perform a legitimate contactless transaction above €25 without PIN, even without the harvesting. However, they state they did not verify their attack because of 'obvious' reasons.

We contacted the authors with our questions. They acknowledged it is difficult to authenticate with an emulated POS terminal, but they think it should be possible. Furthermore, they claim there are no checks in the back end on the correctness of the date or the ATC.

### 4.2.5 Relay Attack with Special Hardware

Kfir and Wool [32] discuss basic design principles of relay attack setups on contactless cards and NFC readers. Their basic relay setup (see Figure 4.3) uses a ghost device that fakes a smart card to the reader and a leech device that fakes a reader to a smart card.

Although Kfir and Wool do not discuss a possible relay attack on EMV Contactless, they state that when the financial gain is high enough, attackers will start virtually pickpocketing their victims. The authors have not implemented a proof-of-concept, but with simulations they give an overview of the achievable maximum distance between smart card and leech device. This overview is presented in Table 4.4. For the 'Standard' method, the authors assume a normal antenna and normal current. For the 'Current + Antenna' method, the authors

| Method | Max Distance in [cm] | Extra Cost beyond NFC in [$] | Availability | Attacker Knowledge |
|---|---|---|---|---|
| Standard | 10 | 0 | High | Low |
| Current + Antenna | 40 | <100 | High | Medium |
| Current + Antenna + Software | 50 | <100 | Medium | High |
| Current + Antenna + Signal-Processing | 55 | >5000 | Low | Very High |

Table 4.4: Leech to tag effort and benefit (copied from [32])

used a stronger current, 4 Ampere, and an optimal antenna for that current. The other two methods ('Current + Antenna + Software' and 'Current + Antenna + Signal-Processing') use the retransmission functionality of the ISO 14443 standard, to request the same message multiple times.

The maximum distance between smart card and fake smart card reader (the leech) is one of the most important limitations when an attacker wants to perform a relay attack on EMV Contactless. Therefore, it is very useful to have these indications of the maximum distance considering the costs, availability of the materials and the necessary attacker knowledge.

### 4.2.6 Relay Attack with Phones

Markantonakis et al. [33] demonstrated a relay attack that uses a BlackBerry phone as card emulating device and an Android device as card reader. With this method, BlackBerry users can simply install an app from the BlackBerry app store, so that anyone with a BlackBerry phone can perform relay attacks (together with someone with an Android device).

However, there are three serious limitations concerning the range, speed and practicality of their method:

- The BlackBerry and Android devices are connected to each other with the wireless short range technology Bluetooth[26]. Bluetooth has a maximum range of 100 meters[27], but typical ranges lie between 20 and 30 meters. This significantly limits the distance between the two devices and cross-border fraud is definitely not possible.

- The authors have not measured the timing performance of relaying EMV cards. However, they measured the performance of a sample transaction

[26] http://www.bluetooth.com/

[27] http://www.bluetooth.com/Pages/Fast-Facts.aspx

consisting of five commands. This sample transaction performed directly with the card takes 152 ms. The time needed for the relay setup with the same card takes 665 ms. The relayed transactions take more than four times as much time, making it very easy for a timing restriction countermeasure to detect this relay attack.

- Once BlackBerry phones were popular and common devices. However, the market share has shrunk to 0.5% in 2014[28]. This low market penetration strongly limits the practicality of this attack.

[28] http://www.idc.com/prodserv/smartphone-os-market-share.jsp

# Chapter 5

# Dutch EMV Contactless System

This chapter describes the EMV Contactless implementation specific to the Netherlands. Section 5.1 describes how the Dutch cards are configured and what features are supported. Section 5.2 describes whether and how the known attacks on EMV apply to EMV Contactless in the Netherlands. Finally, Section 5.3 discusses what the results of a formal analysis on EMV Contactless would be.

## 5.1 Dutch Implementation of EMV Contactless

This section discusses how the Dutch contactless cards are configured with respect to the options, features and parameterizations of EMV Contactless described in Section 3.3. We have examined five different bank cards: an ING bank card from May 2014, an ABN Amro card from May 2014, an ABN Amro card from August 2014, a Triodos card from May 2014 and a Knab card from June 2014. Furthermore, we have also examined two different cards issued by Vodafone, a payment card and a sticker, both from June 2014.

### 5.1.1 Methodology

We used two different approaches to research the behavior of Dutch cards:

- A USB dual interface card reader (Model: Omnikey 5321 v2) and a Python program[29] is used to simulate communication between cards and a POS terminal. With the communication protocol described in Section 3.3, we can infer all the options and parameterizations of the card. We sent multiple commands from the RFID reader to the card and the card responded with the options, parameterizations and card specific data (such as account number and certificates).

[29] kindly provided by Joeri de Ruiter

| Issuer | Brand | Note |
|---|---|---|
| Knab | Maestro | |
| ING | Maestro | |
| Triodos | Maestro | |
| ABN | Maestro | Issuance date August 2014 |
| ABN | Maestro | Issuance date May 2014 |
| Vodafone | Visa | Card |
| Vodafone | Visa | Sticker |

Table 5.1: Available Dutch cards (as of July 2014)

- Two Android devices and two self made applications to capture the communication between POS terminal and card are used. One of the Android devices was configured to emulate a contactless card and was placed to the terminal. The other Android device acted as a reader and was placed to the card. Through a network link the two devices communicated the commands from the terminal to the card and the responses back from the card to the terminal. The application was programmed in a way to capture all communication in a log file. This captured communication of a protocol run is called a trace. Example traces are shown in Annexes A, B, C and D. The details of this setup are extensively described in Chapter 6.

The responses from the card always consist of a command status and response data. If the command is successfully received and supported by the card the command status is `0x9000`, which indicates 'success'. The response data consists of tags, length indicators and content. To further clarify, we describe the first response sent from card to terminal more in depth. The trace snippet is shown in Figure 5.2.

The status `0x9000` indicates that the command is successfully received by the card. Tag `0x6F 2C` indicates that the following 44 bytes ($2C_{(16)} = 44_{(10)}$) contain information in template `6F` ('File Control Information' [11]). This template consists of the Dedicated File Name (of length 15) and another template `A5` (File Control Information Proprietary Template). This `A5` template contains another template `BF0C` (File Control Information Issuer Discretionary Data). This `BF0C` in its turn contains a `61` Application Template for each application the card supports. In this case the card only supports one application so it returns only one Application Template. This Application Template indicates the identifier of the application, the label of the application (MAESTRO) and which priority the application has. The priority is used to select an application when there are more than one supported, so in this case it is not really useful.

We verified the methods and functions that the card claims to support to check if they are really supported, and we tested the functions and methods that

Card → Terminal (message 3 in Figure 3.4):
9000 6F 2C 84 0E 325041592E5359532E4444463031 A5 1A BF0C 17 61 15 4F 07 A0000000043060 50 07 4D41455354524F 87 01 01

9000: Status OK
6F: File Control Information
  84: Dedicated File Name
    325041592E5359532E4444463031: 2PAY.SYS.DDF01
  A5: File Control Information Proprietary Template
    BF0C: File Control Information Issuer Discretionary Data
      61: Application Template
        4F Application Identifier
          A0000000043060
        50: Application Label
          4D41455354524F: MAESTRO
        87: Application Priority Indicator
          01

Figure 5.2: Trace snippet of first response of card to terminal

the card does not claim to support and methods that the card claims to not support.

During inspection of the configuration of the cards, it became clear that some Maestro cards acted in the same way. In particular, the old and new ABN Amro cards, the ING card and the Triodos Bank card acted very similar and the only difference found was dynamic information such as keys and account numbers. The Maestro card issued by Knab Bank, however, acted completely different.

### 5.1.2 Maestro Card Configuration

#### 5.1.2.1 Parameterizations in Card

The card authentication options and whether EMV Mode is supported are indicated by the AIP. The cards all have their AIP set to `0x1980`. This indicates, according to Book C-2 [17, Annex A.1.16]:

- CDA is supported,
- Cardholder verification is supported,
- EMV Mode is supported,
- Terminal risk management is to be performed,

- SDA is not supported,
- DDA is not supported,
- Issuer Authentication is not supported,
- On device cardholder verification is not supported.

The CVM list contains the options for cardholder verification and is set to `0x000000000000000042031F03`. This indicates that [11, Section 10.5]:

- The first cardholder verification rule is "Apply succeeding CVM rule if this CVM is unsuccessful: enciphered PIN verified online if terminal supports this CVM",
- The second cardholder verification rule is "Fail cardholder verification if this CVM is unsuccessful: no CVM required, if terminal supports this CVM".

Translated to plain English this means that the first choice of the card is to verify the PIN online, but if the terminal does not support that, then no CVM should be performed.

The card does not provide a Data Recovery DOL which implies that the recovery of torn transactions is not supported. The card does not provide a Data Storage DOL so Data Storage should not be supported. Absence of the Data Recovery DOL indicates that the card does not support the RECOVER AC command at all[30].

#### 5.1.2.2 Card Differences

We identified two different Maestro applications issued in the Netherlands. The Maestro application found on Knab cards differs significantly from the Maestro application found on ING Bank, ABN Amro and Triodos cards. In the remainder of this thesis, we refer to the application on Knab cards as type A and to the application on the other Maestro cards as type B. The overall behavior is the same, but the type A application has a completely different structure for the responses than the type B application. Furthermore, type A has six available records for the terminal to read while type B has four available records.

A successful transaction with the Knab card is shown in Annex A. The Knab card shows some parameterizations with respect to other tested cards. The following different parameterizations were identified

- Knab cards have Card Transaction Qualifiers (CTQ) specified,
- Knab cards have different application usage controls,

[30] `http://cardconnect.com/uploads/documents/Maestro_Global_Rules.pdf`

- Knab cards have SSAD.

The CTQ is not defined by Maestro specifications. In fact, the only reference to CTQ is made in Visa's Kernel 3 book [18].

#### 5.1.2.3 Card Behavior

The Maestro cards show some inconsistencies between the configuration on the card and the actual implementation:

- All Maestro cards still seem to support DDA. The cards respond as if they support DDA when they receive a `GET CHALLENGE` command which can only be used for DDA.

- The offline PIN verification, however, is supported on the ING card from 2013 while the card indicates in its configuration that it is not supported. This was already discovered by students at the Radboud University Nijmegen[31]. They communicated this to the concerned issuers and in newer cards this functionality has been disabled. A legitimate terminal, however, will never issue an offline PIN verify command, but when a `VERIFY(pin)` command is issued before an application is selected, the older cards respond nonetheless whether the PIN was correct or how many PIN tries there are left. All the newer cards do not support this command. In Book C-2 [17] it is mentioned that offline PIN verification is not considered suitable because of "card in the field timing issues". These issues are not further defined so we can only assume the authors mean that it is inconvenient if the card has to be in the NFC-field for a long time. Indeed offline PIN verification introduces extra messages to and from the card which has a negative impact on the transaction time. However, we do not think this is the biggest issue as Emms et al. describe possible attack scenarios exploiting offline PIN verification on contactless cards in [7].

- It is not possible to use the `READ RECORD` command on record 1 of the data file with the short file ID 1, the place where, if supported, the Mag-Stripe version is normally stored together with the track 1 and track 2 data. However, if a `COMPUTE CRYPTOGRAPHIC CHECKSUM` command is issued to perform a Contactless Mag-Stripe Mode transaction, the card responds as if it supports Contactless Mag-Stripe Mode. This command can only be used for Contactless Mag-Stripe Mode, but PayPass Maestro cards must not support Contactless Mag-Stripe Mode according to the PayPass M/Chip requirements [34]. It remains unclear why the cards support this command, as we cannot see why this would be useful. However, the issuer will likely

[31] Anton Jongsma, Peter Maandag and Robert Kleinpenning

deny these transactions (they are not supposed to be performed). All Contactless Mag-Stripe Mode transactions need to be authorized online but this can be done deferred [34]. Therefore, it is potentially possible to acquire a valid CVC from a contactless card, however, the necessary magnetic stripe data cannot be acquired contactlessly so it is unlikely an attacker can use this valid CVC. For deferred online transactions, the terminal cannot verify whether the CVC (or the magnetic stripe data) is correct, so an attacker would not even need a valid CVC. The terminal will accept any CVC and the merchant only learns the transaction was not successful after the transaction is sent to the issuer.

- The Dutch cards seem to not allow transactions to be performed offline. We tested many combinations of country codes, currency codes and amounts but none of them resulted in the card sending a TC. In fact, any correct request of type TC or ARQC results in an ARQC.

**PayPass Compliancy** All Dutch Maestro cards show the Maestro logo, but none of the Dutch cards show the PayPass logo. It remains unclear why the cards do not show the PayPass logo. It could be the case they are not compliant, as they break for example the following requirement published in 2007[32]: "Maestro PayPass is designed to operate only offline and only with No CVM". This requirement, however, is not present in later versions of the same document[33]. Dutch cards only operate online and also support online PIN. This might mean that the Dutch cards are not PayPass compliant and therefore do not show the PayPass logo. However, the requirements from 2007 seem to be very strict nowadays. Internet is practically available anywhere (at least in the Single Euro Payments Area), and it would be at least a discomfort if customers cannot use contactless transactions anymore for a day once they have payed €50. Therefore, it could be that the cards are in fact PayPass compliant but do not show the PayPass logo for a different reason (e.g., aesthetics or confusion by proliferation of logos).

**Data Storage and Recovery** The card does not provide a Data Recovery DOL or Data Storage DOL and there are no defaults specified for these DOLs. We issued the commands for Data Recovery and Data Storage with some DOLs that are available on the card (e.g., CDOL1 and CDOL2), but we could not get these functions working. Therefore, we can only assume the card really does not support these features.

**Shared Information over Interfaces** Other interesting things we learned about the behavior of Dutch cards include the sharing of the ATC and PIN retry

[32] `https://www.paypass.com/pdf/public_documents/maestroppimpreqjune07.pdf`
[33] `https://www.paypass.com/pdf/public_documents/PPMCAIR(V1.0-July2008).pdf`

counter between the contactless and contact interface. We did not encounter any other dynamic information that was shared between these interfaces. The certificates stored on the card indeed are different for the two interfaces, which indicates that the private keys are different as well. Because the ATC is shared over the two interfaces, we cannot make the card encrypt the same data once over each interface, so there is no way to verify whether the two interfaces use a different secret symmetric key.

**PIN Verification** Cards are configured to do online PIN verify when possible and only do 'no verification' if online PIN is not supported by the terminal. However, when the amount is below €25, contactless transactions are performed without PIN anyway, while the terminal in fact does support online PIN.

In practice in the Netherlands, only the issuer determines whether the PIN should be entered or not. The issuer performs risk management based on the transaction details (amount, country, currency, etc.) and requests for the PIN to be entered on the terminal if necessary. For example, when a 'normal' terminal performs a contactless transaction with an amount above €25, the issuer requests the PIN. However, if the terminal is placed at a motorway toll gate, the issuer will allow transactions up to €100 to be performed without entering the PIN[34].

When the issuer is deciding about PIN verification, the card has already left the field of the reader and actually never learns the outcome of the risk management of the issuer. Because the issuer has the final say on PIN verification, it is rather redundant that the card has these settings configured: they are simply ignored by the issuer.

**Static Commands and Responses** Most commands and responses during a transaction are static (i.e., they are the same for every transaction). Only the `GENERATE AC` command (which contains information such as date, amount and the UN) and its response are dynamic (i.e., they are different for every transaction).

**Minimum Set of Commands for Transactions** From a card's point of view, only a subset of the commands issued during a normal transaction need to be issued for a transaction. This minimum set of commands for transactions consists of selection of the AID, the `GET PROCESSING OPTIONS` command and the `GENERATE AC` command. As a result, the selection of the PPSE does not have to take place and the records do not have to be requested. The cards, however, do not generate an AC unless first the AID is selected, then the `GET PROCESSING OPTIONS` command is issued and then the `GENERATE AC` command is sent.

[34] `http://www.maestrocard.com/gateway/about/about_maestro_paypass.html`

### 5.1.3 Visa Cards Configuration

The Vodafone Visa card has two applications available on its contact interface and two applications on its contactless interface. The Vodafone Visa sticker has no contact interface but also has two applications available on its contactless interface. All applications are labelled V PAY and apparently, there is no difference in the functionality.

The Visa applications only perform online transactions (i.e., they only generate ARQCs and not TCs). Card authentication is not supported. The PDOL contains the following data objects, and thus must be provided by the POS terminal when the `GET PROCESSING OPTIONS` command is issued.

- Terminal Transaction Qualifiers that indicate the capabilities and requirements of the POS terminal,
- The amount of the transaction,
- The UN generated by the POS terminal,
- The currency of the transaction.

The AIP consists of all zero bits. The AIP for Visa is slightly different then for Maestro cards. All zero bits in the AIP indicate that:

- SDA is not supported,
- DDA is not supported,
- Mag-Stripe Mode is not supported,
- The application is not run on a mobile phone.

**Static Commands and Responses** Most commands and responses are static. Only the `GET PROCESSING OPTIONS` command and its response are dynamic. This command contains, among others, the amount and the UN, and is thus different for every transaction. The response to the `GET PROCESSING OPTIONS` command contains the AC generated over the amount and the UN, and is thus also different for every transaction.

**Minimum Set of Commands for Transactions** The cards generate ACs when first the AID is selected and then a `GET PROCESSING OPTIONS` command is issued. The selection of the PPSE can be done before selection of the AID, but does not have to be performed from a card's point of view. The minimum set of commands for a transaction is selection of the AID and the `GET PROCESSING OPTIONS` command.

| Card | Application type | NFC type |
|---|---|---|
| Knab | Maestro type A | NFC type A |
| ING | Maestro type B | NFC type B |
| Triodos | Maestro type B | NFC type B |
| ABN new | Maestro type B | NFC type B |
| ABN old | Maestro type B | NFC type B |
| Vodafone Card | Visa qVSDC | NFC type A |
| Vodafone Sticker | Visa qVSDC | NFC type A |

Table 5.3: All tested Dutch EMV Contactless cards

However, there is a small difference in the response to the `GET PROCESSING OPTIONS` command between the card and the sticker. This difference is discussed in the following paragraph.

**Difference between Card and Sticker** The only difference that we identified between the Vodafone card and the Vodafone sticker applications, is that they have a different 'Form Factor Indicator' field (tag `0x9F6E`). The sticker has value `0x22000000` and the card has value `0x20000000`. The sticker indeed has a different form factor than the card so it comes as no surprise that they are different.

### 5.1.4 Overview

The overview of all tested Dutch EMV Contactless cards with their application type and NFC type is presented in Table 5.3.

## 5.2 Applicability of Known Attacks on Dutch EMV Contactless

In this section all the known attacks on EMV Contact and EMV Contactless are discussed, with a strong focus on their applicability on the Dutch EMV Contactless implementation.

**Cloning Cards with SDA** Technically speaking, SDA is still supported by the contactless specifications. However, we have not encountered a Dutch card which supports SDA, its use is discouraged and so it is unlikely any Dutch card will ever support contactless SDA. Therefore, this attack might be possible for contactless cards in general but it is not applicable to Dutch contactless cards.

**Faking Transactions with DDA** Since DDA is not supported for all MasterCard cards (including the Maestro cards), this attack is not possible for MasterCard or Maestro Dutch cards. Visa Cards also do not support DDA. This attack is thus not applicable to Dutch EMV Contactless cards.

**Eavesdropping** Eavesdropping in most cases is more easy when data is transmitted contactlessly, as no direct contact is necessary and it is possible to hide the eavesdropping device. There is still no end-to-end encryption between card and terminal for EMV Contactless thus all communication can be eavesdropped. This information includes at all times the account number, the amount and the currency, which must all be considered as privacy sensitive information. It is rather trivial to create a device that records all account numbers, amounts, currencies and timestamps of all transactions. This information can be sold or in any other way misused.

However, eavesdropping does not allow an attacker to perform false transactions or clone cards, so the (financial) impact is limited. Nonetheless, from a security and privacy perspective, it is very undesirable that the information of a transaction is communicated plaintext, let alone contactlessly. This really is a major flaw in the EMV Contactless specifications. This shows that the EMV specifications are outdated, do not service the needs and wishes of customers and in our opinion need a thorough and critical revision.

**Relay Attack** The relay attack on EMV Contact described by Anderson et al. [2] can possibly be extended to the EMV Contactless specifications. In some aspects, it may be even easier to perform a relay attack on EMV Contactless cards. Legitimate owners of cards do not have to provide their permission for a transaction (by entering the PIN), so all an attacker needs is a short time in the close proximity of the card. This dramatically decreases the difficulty introduced by the timing of the original attack. Furthermore, the equipment needed is widely available as two Android phones with NFC functionality could suffice. The attack could also be extended to allow transactions for which a PIN is necessary, but then the owner should want to perform a transaction, a fake contactless terminal is needed and timing becomes again crucial.

Indeed it is shown that RFID signals can be relayed between a reader and a card [32, 33]. However, it has only been tested with a sample transaction, and not with an EMV Contactless transaction using a real bank card and a real POS terminal.

In Chapter 6 we extensively describe the proof-of-concept of our relay attack on EMV Contactless.

**Pre-play & Downgrade Attack on Mag-Stripe Mode** The pre-play & downgrade attack on Contactless Mag-Stripe Mode obviously works on some

contactless cards as shown by Roland and Langer [40]. However, this attack will likely not work on the currently available contactless cards in the Netherlands. We did not encounter any card that had Mag-Stripe Mode enabled. However, the functionality is still present as Mag-Stripe Mode transactions can be performed. This can easily be recognized by the issuer as soon as it receives the request to process a Contactless Mag-Stripe Mode transaction, since it should not expect such transactions. Furthermore, it is not known whether Dutch terminals support Contactless Mag-Stripe Mode, since there is not really a need for that. However, if terminals support the Contactless Mag-Stripe Mode and issuers do not perform adequate checks for this, the attack might be applicable to the Dutch EMV Contactless implementation.

**Offline Verify PIN Attack** Of all the Dutch contactless cards, to our knowledge, only the first ING Bank and ABN Amro Bank cards supported offline PIN verification. This presumably is the result of an error. The card's AIP indicates that support for offline PIN verification is not available (just as it should be configured, according to the specifications). However, for some reason the functionality is still available if it is issued anyway. One plausible explanation is that the code for the contact interface was copy-pasted, the AIP was changed for the new configuration but the 'old' function for PIN verification was not truly removed from the code. For these cards, this attack might be possible, however, there are a few remarks that greatly reduce the impact of the attack. The authors of [7] estimated the chanche of guessing a customer-chosen PIN correctly with six guesses at 8.23%. However, with these ABN and ING cards, the chance that a correct PIN is quickly guessed correctly is much smaller than 8.23%, because only two guesses can be made without blocking the card. Furthermore, the majority of these first generation contactless cards are issued by ING Bank, which does not allow customers to change or choose their PIN. As a result, there is no correlation between significant dates or other personal numbers and the PIN for ING cards. With this random distribution of PINs, one would need 5000 guesses (or 2500 access moments) to have a chance of 0.5 to guess it correctly. This does not look like a realistic attack scenario, however, a small subset of ABN Amro contactless cards might be vulnerable.

Furthermore, when an attacker wants to guess PINs more than once for the same card, he might want to be even more cautious. Most POS terminals, including random readers (see Figure 5.4), give a warning after the PIN has been entered two times incorrectly. This will almost definitely raise suspicion for card owners if this keeps occurring. We verified that the PIN retry counter does not reset after a successful contactless transaction for Dutch cards for which the PIN was not entered. This does not come as a surprise: the card cannot even know whether the contactless transaction was successful or not.

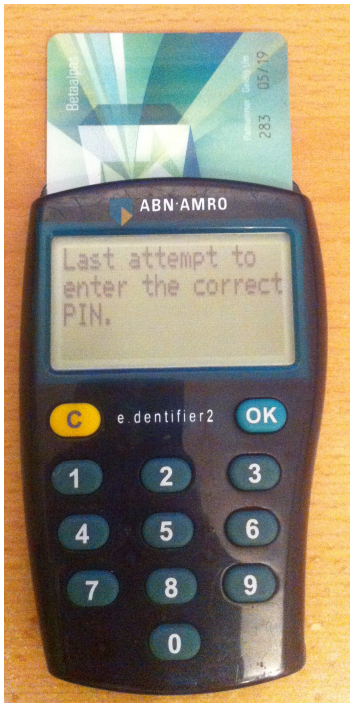


Figure 5.4: Random reader giving warning after two invalid PIN tries

**Harvesting High Value Foreign Currency Transactions** As discussed in Section 4.2.4, we already have serious doubts whether this attack is possible in the UK. It is definitely not possible in the Netherlands, as none of the Dutch cards allow offline transactions. Furthermore, Dutch terminals do not allow offline transactions and there is a hard limit of €25, or equivalently in foreign currency, for transaction without a PIN code. Therefore, an attacker cannot use a Dutch card for the harvesting and cannot use a Dutch rogue merchant account for clearing the harvested TCs.

## 5.3 Formal Analysis

Our initial plan was to extend the work of De Ruiter and Poll [5], which describes a formal analysis of the EMV Contact specifications. Three known attacks on the communication protocol of EMV Contact have been found again with the formal analysis. However, during our research on EMV Contactless in general and the Dutch implementation specifically, it appears a formal analysis of the Dutch EMV Contactless would be rather meaningless. The Dutch EMV Contactless implementation is mere a subset of the EMV Contact specifications analyzed by De Ruiter and Poll, rather than an extension. SDA and DDA are no longer supported and offline PIN verify is disabled for contactless transactions. Fur-

thermore, the only possible contactless transaction is one which uses CDA and which is authorized online. From the known attacks, one attack exploits SDA, one attack exploits DDA and the third attack exploits offline PIN verify. These two card authentication and cardholder verification methods are no longer supported. Therefore, the found attacks on EMV Contact during the formal analysis are not applicable to the Dutch EMV Contactless implementation. In addition, the Dutch EMV Contactless implementation does not offer any new functionality which could possibly allow a new vulnerability. Therefore, we have refrained from making a formal analysis of the Dutch EMV Contactless implementation as the results can already be logically derived from the results as presented by De Ruiter and Poll: No attacks will follow from a formal verification of the communication protocol of the Dutch EMV Contactless implementation.

The attacker model for EMV Contactless, however, is somewhat different from the attacker model for EMV Contact. With EMV Contactless, it is much more easy for an attacker to eavesdrop or manipulate data due to the contactless interface. For EMV Contact this is much harder and specialized and miniaturized technology is needed. However, De Ruiter and Poll used the Dolev-Yao attacker model, which is more powerful than both the attacker models for EMV Contact and EMV Contactless. Therefore, the different attacker model for EMV Contactless will not change the outcome of their formal verification.

# Chapter 6

# Proof-of-Concept Relay Attack

This chapter describes the goals and details of our proof-of-concept relay attack. We present the methodology and the setup we used for the proof-of-concept, and the tools we developed to measure the results. The results are presented in Chapter 7. Chapter 8 describes the other vulnerabilities that are identified and Chapter 9 describes the most likely attack scenarios. Chapter 10 describes the encountered countermeasures and includes the proposed countermeasures from the point of view of the issuers, terminal manufacturers and customers. All results, identified vulnerabilities and attack scenarios and countermeasures are then discussed in Chapter 11.

## 6.1 Relay Attack

With a relay setup, the POS terminal is tricked in thinking it is acting directly with a card, while the card is tricked in thinking it is acting directly with a POS terminal. In reality, both card and terminal each act with a different device. Doing this, an attacker can possibly make a transaction with the payment card of a victim. This card can still be in the wallet of the victim and the victim does not have to be close to the POS terminal.

In practice, it is hard to detect a relay attack, as the content of the communication is legitimate. The most obvious countermeasure is to measure the delay of responses to detect overhead introduced by relay setups. If the measured delays do not comply with the expected delays, one of the two legitimate devices can reject further communication.

**Overview** The general overview of a relay setup is shown in Figure 6.1. A relay setup typically uses a Relay Device and a Mole Device. The Relay Device is put close to the POS terminal and the Mole Device is put close to the card. The POS terminal sends commands to the Relay Device, which forwards these commands to the Mole Device. The Mole Device sends them to the card and

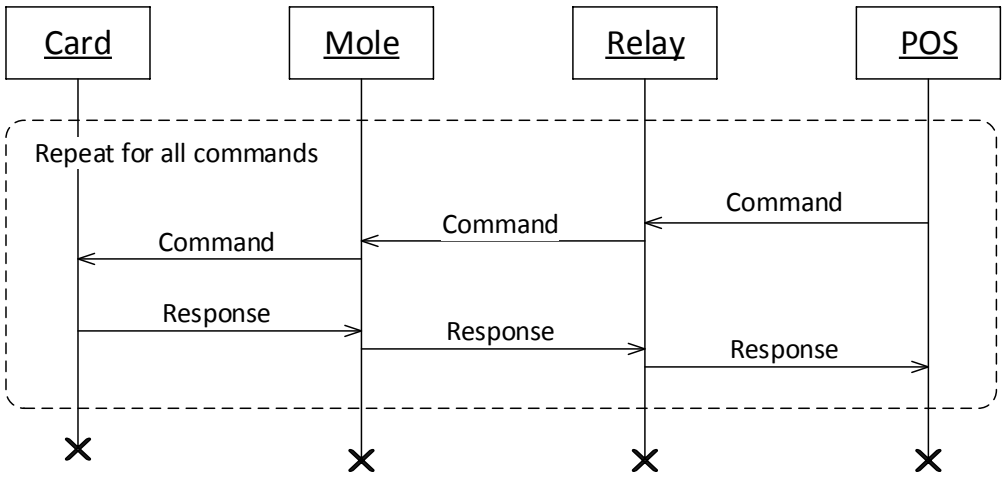


Figure 6.1: General message sequence diagram for the Basic Relay Setup

receives the response. The response is then forwarded via the Relay Device to the POS terminal.

**Goal of the Attack** The goal of our relay attack is to perform a payment with a POS terminal using a card that is not close to the POS terminal. Additionally, the goal is to achieve this with standard mobile devices, without special hardware or a custom operating system. All countermeasures need to be circumvented. In practice, this means the delay needs to be small enough so that the attack is not detected by the card or reader.

Depending on the connection of the Mole and Relay Device, the range can be dramatically increased from approximately 10 cm for NFC to a worldwide range with the use of the Internet. This is a serious threat since the card of an unsuspecting victim can be used to pay for the goods of an attacker. Because the card can communicate contactlessly, it does not have to leave the pocket or wallet of the victim.

Although ING Bank[35], ABN Amro[36] and Knab[37] all independently claim transactions can only be performed when the card is in the close proximity of the POS terminal, the goal of our proof-of-concept is to make a transaction with a contactless card in a wallet in a pocket of a victim.

[35] https://www.ing.nl/particulier/klantenservice/veelgestelde-vragen/betalen/betalen/contactloze-betaalpas/algemeen/index.aspx

[36] https://www.abnamro.nl/nl/prive/betalen/betaalpas/uw-nieuwe-betaalpas.html

[37] https://www.knab.nl/contact/veelgestelde-vragen/betalen/contactloos-betalen

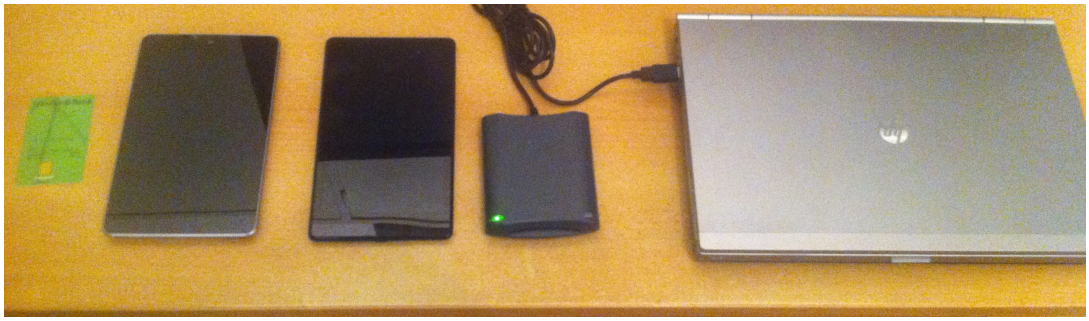

Figure 6.2: Our test relay setup: (from left to right) bank card, Mole Device, Relay Device, emulated POS terminal)

## 6.2 Methodology

The global setting used for our proof-of-concept relay attack is the following: one legitimate POS terminal, one Relay Device which is placed close to the POS terminal and acts as a card, and one device that acts as a POS terminal and is placed close to a legitimate card. Our setup is shown in Figure 6.2.

When a connection is made between the Relay and Mole Device, communication from the POS terminal can be relayed to the card and back again through the Relay Devices. This way, the POS terminal 'thinks' it is communicating directly to the card and the card 'thinks' it is communicating directly to the POS terminal. This way, payments can be made where the card is not close to the POS terminal. When the Mole Device is placed close to a pocket or a bag with a card of a victim in it, an attacker can perform a payment with the account of the victim.

Relaying messages over an extra channel (the connection between Mole and Relay Device) will likely add an extra delay to the interaction. The most obvious countermeasure against such an attack is thus to measure the timing of a transaction, and decline it if it deviates too much from the expected value. Therefore, we are particularly interested in the following characteristics:

- The time a POS terminal allows for a transaction to complete,
- The time a normal transaction takes,
- The fastest achievable transaction time with a relay attack.

The time a POS terminal allows for a transaction to complete cannot be too close to the time a normal transaction takes, otherwise the POS terminal would decline legitimate transactions for small variations in the transaction time. Therefore, we also investigate potential factors that affect transaction times, such as whether the card is in a wallet.

## 6.3 Equipment

This section describes the process of equipment selection and the configuration of the four components needed fro the relay attack.

One of the equipment choices is to use NFC-enabled mobile devices for Relay and Mole Devices. In this way, no knowledge of hardware and antennas is necessary for the NFC connection and connecting two mobile devices through Internet is also relatively straightforward. When Android mobile devices are used, applications that use the NFC functionality can be programmed in the widely supported and documented Java programming language. Furthermore, Android devices have a large market share. Relay apps that can be distributed via the Android Market, dramatically increase the impact as practically anybody could then perform such an attack.

The first Android device with NFC support was the Google Nexus S released in 2010. The NFC functionality, however, was limited as the device could only act as a reader and could not emulate a smart card. For a relay setup, however, one of the Android devices needs to act as, or emulate, a smart card.

This card emulation functionality is called Host-based Card Emulation (HCE)[38] and was introduced in Android 4.4 in October 2013. HCE does not work with every NFC chip available in Android devices. For example, it was not supported in our Google Nexus 7 model 2012. We could not find a definitive list with devices that support HCE, but certain HCE enthusiasts have created such a list[39]. Unfortunately, some models appear to have been released with different NFC hardware, so that some devices of one model support HCE while other devices with the same model name do not support it. There are claims that all Broadcom[40] NFC controllers support HCE, while only a subset of NXP[41] NFC controllers supports HCE.

### 6.3.1 POS Terminal

NFC enabled POS terminals are not cheaply available. Prices start at around €500 and a two-year contract is mandatory. Therefore, we programmed our own POS terminal in Java. In combination with an Omnikey 5321v2 contactless USB reader[42], a POS terminal is emulated. This way, the entire communication between card and reader is simulated.

The emulated POS terminal acts exactly as a real POS terminal, with the exception of the generation of the UN (for our purposes we do not need a secure

[38] http://developer.android.com/about/versions/kitkat.html#44-hce

[39] http://stackoverflow.com/questions/22237583/list-of-devices-support-hce

[40] http://www.broadcom.com/products/NFC

[41] http://www.nxp.com/

[42] https://www.hidglobal.com/sites/hidglobal.com/files/resource_files/omnikey-5321-v2-usb-reader-en-ds.pdf

random number generator, a static 'UN' also suffices). Furthermore, the bank card performs a transaction with the emulated POS terminal identically to the way it performs a transaction with a real POS terminal.

A self programmed POS terminal is much more flexible than a regular POS terminal. We can see how the card reacts to variants in the commands and we can easily measure timings for commands, responses and transactions. We do not want banks to process the transaction thus we do not send the ACs to the bank. This way, so we are free to experiment without raising any fraud alarms in the back end system of the banks.

Our Java terminal detects whether a Visa card or a Maestro card is presented and follows one of the two following protocols:

- For Maestro cards, the communication protocol is shown in Figure 3.4 and messages 1 and 4 in Figure 3.6 (message 5 and 6 are omitted because we do not want the bank to process the transaction). The card sends an AC in the last message of the communication which cannot be decrypted, but the card also indicates whether the transaction was successful by setting the CID. In a normal transaction, the card never learns whether the POS terminal sends the AC to the bank and whether the bank accepts the AC, so from a Maestro card's point of view, the transaction is both valid and final upon sending a valid AC.

- For Visa cards, the communication is shown in Figure 3.7. The communication to the bank is omitted because we do not want the bank to process the transaction. Visa cards also never learn whether the POS terminal sends the AC to the bank and whether the bank accepts the AC, so also from a Visa card's point of view, the transaction is both valid and final upon sending a valid AC.

### 6.3.2 Relay Device

We purchased a Google Nexus 7 model 2013 device as Relay Device. Based on experiences found on forums, we were relatively sure the 2013 model would support HCE. At the moment of purchase (July 2014), it was the cheapest Android 4.4 device with HCE capable NFC hardware for €182.

The Relay Device can connect to the Mole Device when the Mole Device has created a listening server. All network interaction is initiated by the Relay Device. When the Relay Device is placed close to a POS terminal, it receives commands. These commands are sent immediately over TCP/IP to the Mole Device. After sending, the Relay Device waits for a response from the Mole Device. When it receives the response, it is immediately returned to the POS terminal. The POS terminal can now process the response and transmit the next command. Android's HCE implementation requires that the used AIDs are

| Card | # Commands per Transaction | # Commands in Minimum Set |
|---|---|---|
| Knab | 11 | 3 |
| ING | 8 | 3 |
| Triodos | 8 | 3 |
| ABN new | 8 | 3 |
| ABN old | 8 | 3 |
| Vodafone Card | 3 | 2 |
| Vodafone Sticker | 3 | 2 |

Table 6.3: Analyzed cards

declared by the app. Therefore, when Android receives a 'Select AID' command, it can forward the response to the appropriate app. As a consequence, whenever Android receives another AID that is not declared by the app, the commands are no longer forwarded to that particular app. Therefore, we associated the '2PAY.SYS.DDF01' (PPSE), Visa Electron, Visa V PAY and Maestro AIDs with our app.

### 6.3.3 Mole Device

For our Mole Device we used a Google Nexus 7 model 2012 with Android 4.4, which does not support HCE. The Mole Device can start a server to listen on a specific TCP/IP port. The Relay Device can then connect to the Mole Device. It notifies the user when a compliant smart card enters its proximity. The behavior of the Mole Device is dependent of the different relay setup that is used and is further discussed in more detail in Sections 6.4.1, 6.4.2 and 6.4.3.

### 6.3.4 Smart Card

To get a complete overview of all cards and timings, we performed our tests on all EMV Contactless cards that were available in the Netherlands in July 2014. Table 6.3 shows these cards, there specifications and other characteristics.

## 6.4 Relay Setups

This subsection describes the three different relay setups that we have created. The first one is the most basic, with no EMV specific optimizations in the protocol to make the attack faster. This Basic Relay Setup is described in Section 6.4.1. An attacker, however, might want to optimize the time needed to present the Relay Device to the POS terminal (the transaction time), for example, when the

POS terminal has a time based relay attack detection. This Transaction Time Optimized Relay Setup is described in Section 6.4.2. In another scenario, an attacker might want to optimize the time needed for the Mole Device to interact with the victim's card (the coupling time), for example when he does not want to raise suspicion of the victim. This Coupling Time Optimized Relay Setup is described in Section 6.4.3.

### 6.4.1 Basic Relay Setup

For the initial setup we used the Basic Relay Setup shown in Figure 6.1. Every command and response is relayed back and forth between Relay and Mole Device. For cards with Maestro application type A eight commands are issued (see Annex A). For cards with Maestro application type B ten commands are issued (see Annex B) and for cards with the Visa application only three commands are issued (see Annex C).

The transaction starts as soon as the POS terminal sends the first message to the Relay Device. It is not necessary for the Relay and Mole Device to be connected at this time, although it dramatically increases the transaction time. Otherwise, if they are not already connected, the Relay Device waits for this to happen before relaying the command received from the POS terminal. Furthermore, strictly speaking, the Mole Device does not need to be connected to the bank card. The Mole Device can wait with sending the command (received via the Relay Device from the POS terminal) until it is connected to a card, but this again dramatically increases transaction time. Therefore, it is better if the Relay Device is tapped to the POS terminal after it is connected to the Mole Device and after the Mole Device is connected to the bank card.

**Relay Device** For the Basic Relay Setup, the Relay Device simply relays all commands that it receives from the POS terminal, according to Figure 6.1. The commands are sent to the Mole Device over TCP/IP. As soon as the Relay Device receives a response from the Mole Device, it forwards this response to the POS terminal.

**Mole Device** When both the Relay Device has connected to the Mole Device and the Mole Device has an EMV Contactless card in its proximity, the Relay and Mole Devices go in operational mode. In operational mode, the Mole Device waits to receive a command from the Relay Device over TCP/IP. Whenever it receives such command, it transmits this command immediately to the smart card. The smart card sends its response to the Mole Device back which forwards the response immediately to the Relay Device over TCP/IP. Then the Mole Device again waits for the next command from the Relay Device.

### 6.4.2 Preloaded, Transaction Time Optimized Setup

With the Basic Relay Setup, every command and response pair is transmitted twice (once for the Mole Device and card, and once for the Relay Device and the POS terminal). We improved our model to minimize the number of relayed messages. In Sections 5.1.2.3 and 5.1.3 it is mentioned that per transaction, only one command and only one response is dynamic, while the rest of them are static. Furthermore, when the minimum set of commands needed for a transaction is used in stead of the set of commands a terminal uses, the number of commands issued can be reduced from eleven to three for Maestro type A cards, from eight to three for Maestro type B cards, and from three to two for Visa cards (see Sections 3.4.3 and 3.5).

We have revised our relay setup to exploit these properties. The first phase is the preloading phase, shown in Figures 6.4 and 6.5. This phase is only needed once for each individual bank card. The second phase is the actual preloaded transaction phase, shown in Figures 6.6 and 6.7.

Because some of the responses are static, we can preload the Relay Device with these responses before we start the transaction. These preloading protocols are shown in Figure 6.4 for Maestro and in Figure 6.5 for Visa cards. With these protocols, the Mole Device can acquire all static responses from a bank card before the start of the actual transaction.

The transaction begins again when the Relay Device is placed close to the POS terminal. The transaction speed again dramatically increases when at this point there is no connection between Relay Device and Mole Device, or between Mole Device and bank card. This protocol is designed to be fast, so it does not make much sense to start a transaction when not all devices are connected, yet it is possible to let the POS terminal wait.

The Mole Device now sends all static responses to the Relay Device, which puts the static responses in its cache. When a transaction is started by a POS terminal, the Relay Device can now respond to the static commands with responses from its cache, eliminating the need to send these commands over the network. These protocols are shown in Figure 6.6 for Maestro cards and in Figure 6.7 for Visa cards. As soon as a POS terminal starts a transaction with the Relay Device, the Relay Device sends a message to the mole to get ready for a transaction. While the Relay Device is giving cached responses to the POS terminal, the mole can start the transaction simultaneously, up until the point where the dynamic command is needed.

When the Relay Device receives the dynamic command (`GENERATE AC` for Maestro or `GET PROCESSING OPTIONS` for Visa) from the POS terminal, it forwards this command to the Mole Device. The Mole Device at this stage is already done with preprocessing the transaction and is waiting for the dynamic command. As soon as it receives this command, it transmits it to the card. The response (the AC) is sent via the Mole Device through the Relay Device to the POS terminal.

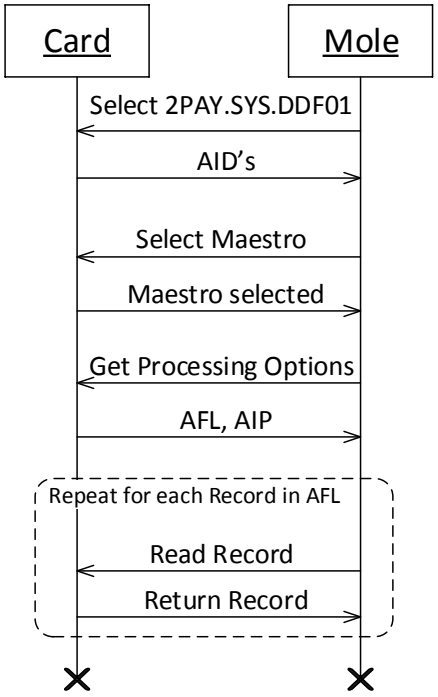


Figure 6.4: Preloading protocol for Maestro cards

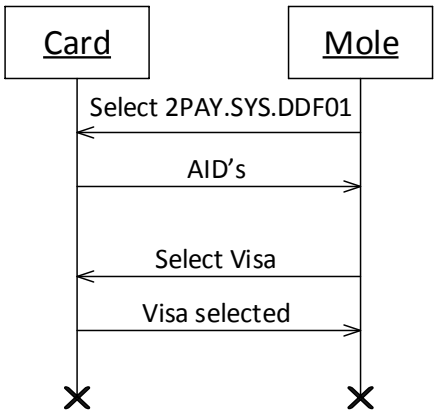


Figure 6.5: Preloading protocol for Visa cards

The Mole Device only has to issue the minimum set of commands for a transaction, described more in depth in Sections 5.1.2.3 and 5.1.3. Therefore, it is likely that the Mole Device is already done with the static part of the transaction

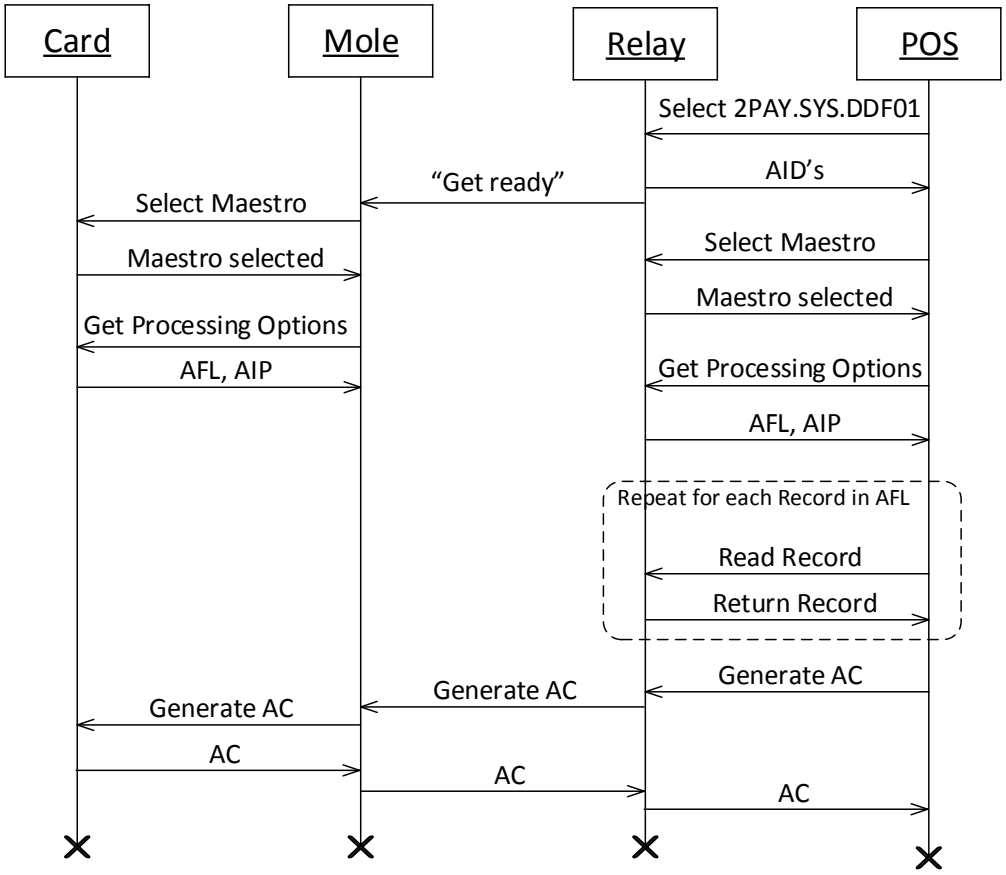


Figure 6.6: Transaction Time Optimized Relay Setup protocol for Maestro cards after preloading

when the Relay Device sends the dynamic command.

With this setup, the Relay Device has to wait only once at a response from the Mole Device (when sending the dynamic command). The added transaction time for this relay setup is mainly determined by the time needed for one network message to reach the Mole Device and for one network message to reach the Relay Device, and the time the Relay Device needs to transmit the response from the Mole Device.

The AC needs to be generated over (among others) the UN from the terminal sent with the dynamic command for transactions to be valid. Only one dynamic command and the corresponding dynamic response are sent from and to the Relay Device. Furthermore, all non-necessary commands are omitted to ensure that the Relay Device does not have to wait for the Mole Device to be ready to process the dynamic command. Therefore, this protocol is the best that we can do to

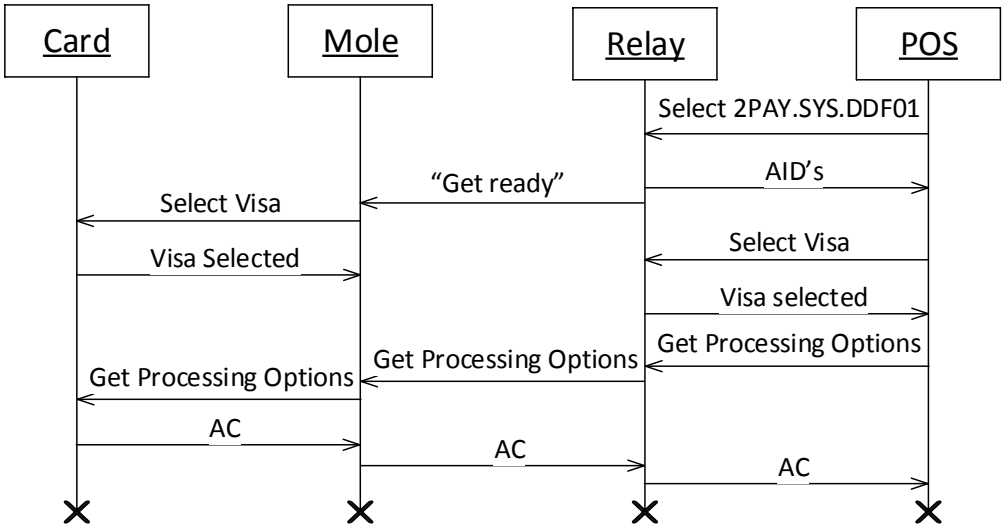


Figure 6.7: Transaction Time Optimized Relay Setup protocol for Visa cards after preloading

minimize the transaction amount.

### 6.4.3 Preloaded, Coupling Time Optimized Setup

In the previous described setup, the time for which the Relay Device is presented to the POS terminal is optimized. In some specific scenarios, an attacker might want to minimize the interaction time between the Mole Device and card (the coupling time). Doing so, he does not raise suspicion with the victim by staying too close for too long. In this case, the protocol needs to be designed in a way so that the Mole Device does not have to wait for the Relay Device.

These protocols are shown in Figure 6.8 for Maestro cards and in Figure 6.9 for Visa cards. The Relay Device starts the transaction, using the static commands from its cache, obtained with the preloading protocols shown in Figures 6.4 and 6.5. The Relay Device proceeds until the point where it receives the dynamic command. This dynamic command is forwarded to the Mole Device. Until this point, there was no interaction with the card and the card does not have to be in the proximity of the Mole Device. After these actions, the Mole Device has all necessary information to complete the transaction from a card's point of view. When the smart card then enters the proximity of the Mole Device, the Mole Device performs the minimum set of commands to perform a transaction with the already received dynamic command. The response of the card to the dynamic command (the AC) is directly relayed to the Relay Device, which can then

transmit it to the POS terminal.

The transaction begins when the Relay Device is placed close to the POS terminal. This protocol is designed for a short coupling time, so the Mole Device does not need to be connected to the bank card yet. After the mole received the dynamic command via the Relay Device from the POS terminal, it can be placed close to a card. Now the transaction begins from the card's point of view.

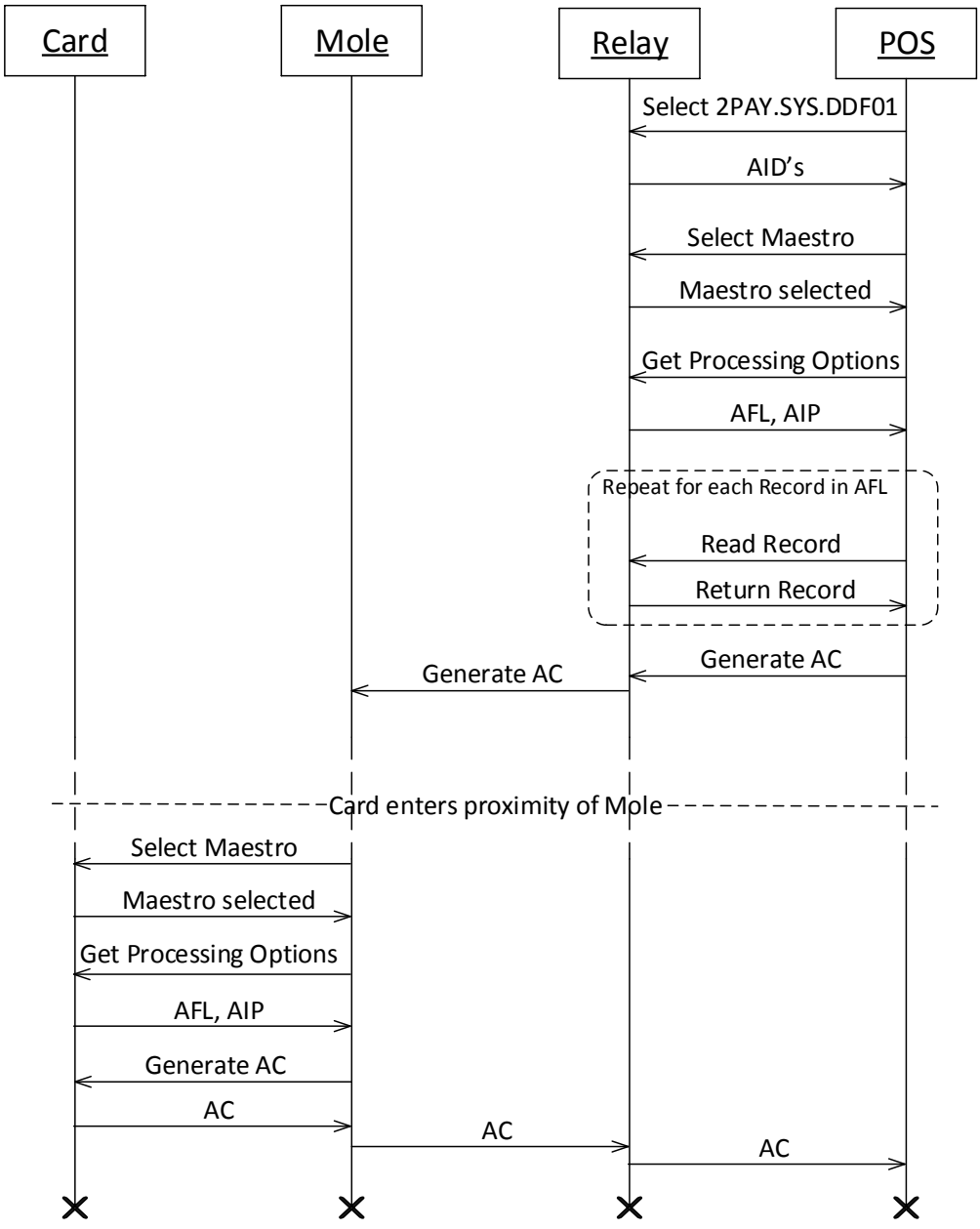


Figure 6.8: Coupling Time Optimized Relay Setup protocol for Maestro cards after preloading

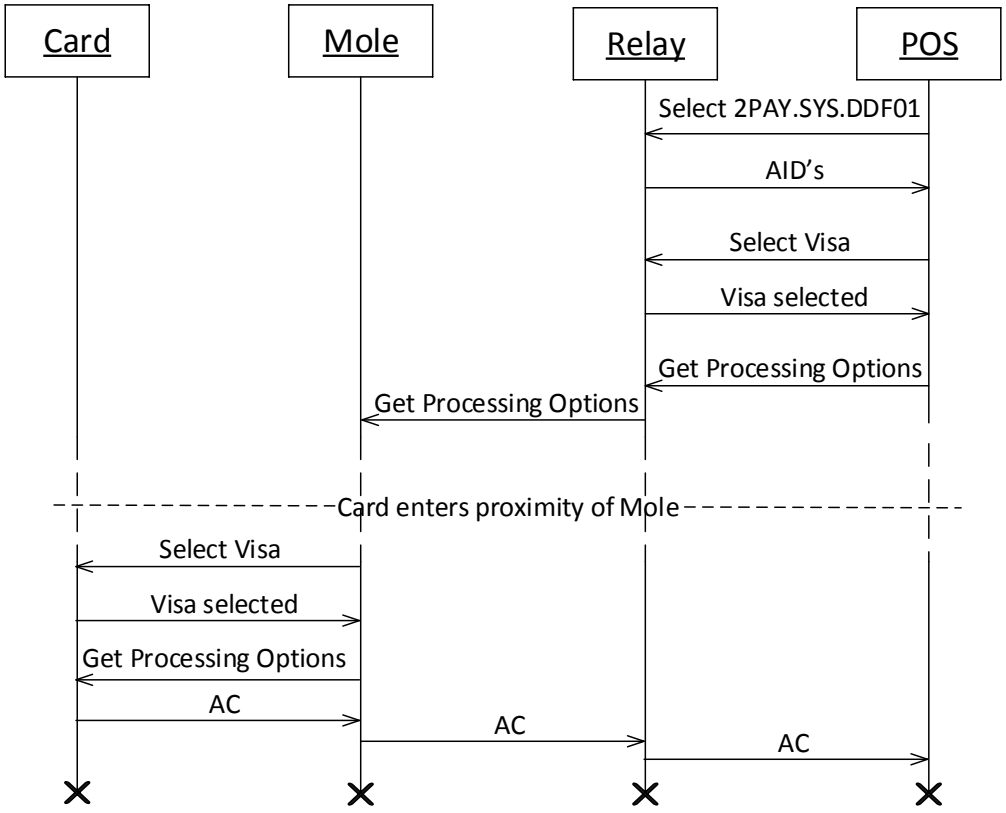


Figure 6.9: Coupling Time Optimized Relay Setup protocol for Visa cards after preloading

# Chapter 7

# Relay Setup Performance

This chapter first presents the performance results of the four different components of our relay attack. Furthermore, it presents the performance results of the different relay setups as described in Chapter 6. Together with other identified vulnerabilities in Chapter 8, we have identified several potential attack scenarios. These attack scenarios are described in Chapter 9.

Section 7.1 gives a detailed overview of the remainder of this chapter.

## 7.1 Timing Measurements Overview

All seven cards from Table 5.1 are used for measuring the card timings. For the POS terminal results, we have used an emulated POS terminal and an Atos Wordline Yoximo[43] POS terminal. The Relay Device results are measured with a Nexus 7 model 2013 in HCE mode. The Mole Device results are measured with a Nexus 7 model 2012 in NFC reader mode. The network delays are measured with the two Android devices connected through a standard wifi router.

Section 7.2 presents the timings of all the cards tested with the emulated POS terminal. These timings include the delays for individual command and response pairs and total transactions. These results give a good indication about the timing ratios between different commands for the same card. Furthermore, it provides insight in how the different cards perform with the same commands. The different delays between cards is essential to determine how successful a relay attack can be. If all delays are equal for all cards, a relay setup can easily be detected. However, if there is a significant difference between the timings, the terminal might not be able to distinguish between a transaction performed directly with slow card and a relayed transaction with a fast card. Furthermore, if there are small differences between individual commands, then these small differences could add up to a more significant difference for complete transactions. As a result,

[43] `http://www.banksys.com/adminV3/ContentManager/display/000/509/345/5093451.pdf`

relay setups can also be detected by measuring the complete transaction timings, so these are discussed as well.

Section 7.3 presents results which compare the timings between the emulated POS terminal and a real POS terminal. With this comparison, an assessment is made about the performance of real POS terminals. Furthermore, the timing restrictions of the real POS terminal are also discussed. Section 7.4 presents the timing delays for communication between the Relay Device and the POS terminal. Section 7.5 presents the timings of all cards in combination with the Mole Device. Card delays are higher and less consistent with the Mole Device than with the emulated POS terminal. The results, measured with the Mole Device, give a better insight in how well the relay attack performs, because real world attacks are performed with resource-constraint Mole Devices, and not with fast computers. Section 7.6 gives insight in the added network delay between the two Android devices. Finally, Section 7.7 describes the performance results of the different relay setups.

## 7.2 Card Timings

One of the critical aspects for a successful relay attack is to know how much time a legitimate transaction takes. The timings in this section are measured with our self programmed POS terminal Java application. One instruction before the program transmits a command, a timer is started. The timer is stopped the moment after the corresponding response is received. The Java application is run on a fast computer, so that the reader is not the significant bottleneck of the setup. However, this computer still uses Java as programming language and a USB interface for communication with the reader. Therefore, cards could perform even faster on different hardware, e.g., a dedicated NFC reader without USB interface or Java layer.

With this method we get a good idea of how long such a command and response would take on a real POS terminal. A significant timing difference between different cards makes it easier for attackers to perform relay attacks. POS terminals typically do not have knowledge about the expected timing, so they should allow at least all timings that are below the maximum timing of the slowest card.

The timing results per command of Maestro cards are shown in Table 7.1. Table 7.2 shows the timings per card per specific record that is read by the POS terminal. The results of Visa cards are shown in Table 7.3.

Each card performed ten transactions and we measured the timings. As the timings for a command are very consistent per card (typically 1 or 2 milliseconds between minimum and maximum timing), ten transactions is enough to get a good idea about the timings. We present the lowest measured timing (min), the most occurred timing (the mode), and the highest measured timing (max). The

| | | ING | Triodos | ABN new | ABN old | Knab |
|---|---|---|---|---|---|---|
| Select PPSE | min | 26 | 25 | 25 | 39 | 15 |
| | mode | 26 | 25 | 25 | 39 | 16 |
| | max | 26 | 25 | 26 | 40 | 16 |
| Select AID | min | 24 | 24 | 24 | 39 | 15 |
| | mode | 24 | 24 | 25 | 40 | 15 |
| | max | 25 | 25 | 25 | 40 | 32 |
| Get Processing Options | min | 37 | 36 | 36 | 45 | 15 |
| | mode | 37 | 36 | 37 | 46 | 31 |
| | max | 37 | 37 | 38 | 46 | 32 |
| Read Records | min | 94 | 93 | 94 | 113 | 106 |
| | mode | 96 | 95 | 96 | 118 | 125 |
| | max | 99 | 97 | 98 | 119 | 144 |
| Generate AC | min | 174 | 171 | 173 | 331 | 154 |
| | mode | 174 | 171 | 174 | 332 | 154 |
| | max | 175 | 172 | 175 | 332 | 175 |

Table 7.1: Timing results of Maestro commands per smart card in [ms]

mode of the measured timings is more relevant than the average value, because the mode gives insight in a typical transaction. The average value can be greatly influenced by only one single high measurement in which case this particular average value may not even occur once during the measurements.

Tables 7.1 and 7.3 show that the bank cards are rather consistent in their timings, in contrast to the Nexus 7 emulating a card (as shown in Table 7.5). The results show that the Knab card is the fastest of the Dutch Maestro cards. An interesting observation, however, is that reading the records takes more time on a Knab card than on cards that are otherwise slower. We think this is mainly because the Knab card has more and longer records (see Annexes A and B). The old ABN Amro card is significantly slower than all the other Maestro cards for all commands. Both Vodafone Visa cards are equally fast.

Figure 7.4 shows the normal transaction times and the transaction times when only the minimum set of commands is used (as discussed in Section 5.1.2.3 and 5.1.3). Only the timings of complete transactions are shown, and timings of individual command and response pairs are not included.

| | Knab | ING | Triodos | ABN new | ABN old |
|---|---|---|---|---|---|
| SFI 1, Record 1 | 16 | - | - | - | - |
| SFI 1, Record 2 | - | 11 | 10 | 11 | 14 |
| SFI 1, Record 3 | - | 33 | 33 | 33 | 39 |
| SFI 2, Record 1 | 15 | - | - | - | - |
| SFI 3, Record 1 | 31 | 22 | 22 | 22 | 27 |
| SFI 3, Record 2 | 16 | 30 | 29 | 30 | 35 |
| SFI 4, Record 1 | 16 | - | - | - | - |
| SFI 4, Record 2 | 31 | - | - | - | - |

Table 7.2: Timing results of reading records per smart card in [ms]

| | | Vodafone Sticker | Vodafone Card |
|---|---|---|---|
| Select PPSE | min | 29 | 28 |
| | mode | 29 | 28 |
| | max | 30 | 29 |
| Select AID | min | 45 | 30 |
| | mode | 46 | 30 |
| | max | 46 | 31 |
| Get Processing Options | min | 101 | 97 |
| | mode | 102 | 98 |
| | max | 102 | 103 |

Table 7.3: Timing results of Visa commands per smart card in [ms]

**Additional Delay for First Transaction** Another interesting observation is that the first transaction of Maestro type B cards is significantly slower than each subsequent transaction. Typical transaction times for ING Bank, Triodos and the new ABN Amro card are 360 milliseconds, while the first transaction is approximately 100 milliseconds slower. We found out that this 100 milliseconds extra delay for a transaction disappeared when there was some communication between card and reader before the transaction. Upon further inspection, it appeared that this extra delay was mainly introduced by a significantly longer response time for the first response of the card and a slightly longer response time for the second response of the card. The third and subsequent response times were not influenced by this behavior. We suspect that the RFID protocol used by these Maestro type B cards is the reason behind this: the protocol may need extra time

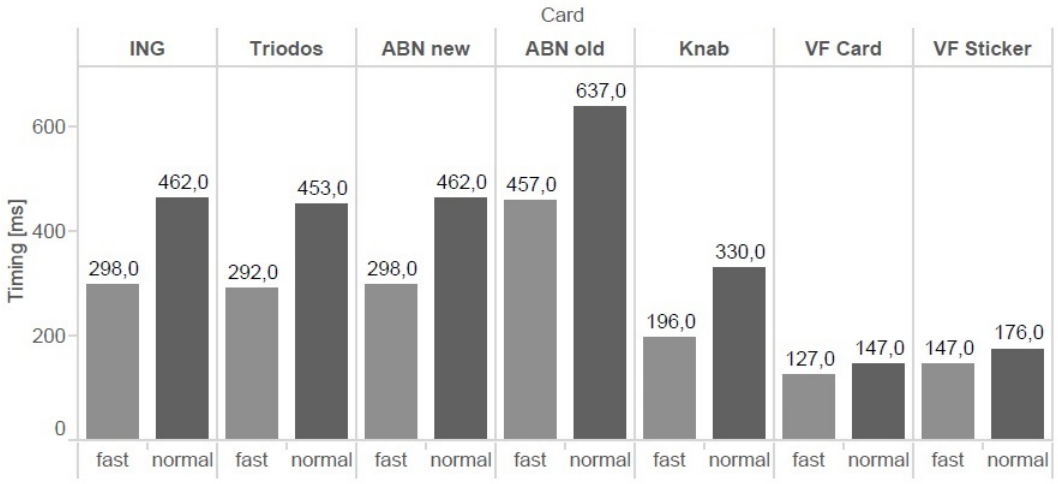


Figure 7.4: Transaction times for normal transactions and transaction using only the minimum set of commands

for initialization, handshaking or parameter agreement. Furthermore, we think the initialization is only completed when the first commands are exchanged, not upon detection of the card.

The Maestro type A card and the Visa cards do not show this behavior. This coincides with the different RFID protocols that the cards use: All cards that suffer from the initialization delay are of type ISO-14443-B while all cards that do not suffer from this delay are of type ISO-14443-A.

However, we have no proof that this behavior is a consequence of different protocols or different types defined in the ISO standard. The cause of this behavior is very likely to be found in a low-level implementation of a layer in the NFC communication stack, and has nothing to do with EMV. Therefore, we do not investigate this any further. This behavior is advantageous for attackers who perform relay attacks. Timing restrictions cannot be stricter than the time needed for the slowest cards, which means attackers have a larger time window for relaying transaction with fast cards.

**Additional Delay for Card-in-Wallet Transactions** All previously mentioned timings are from transaction performed with the card directly placed on the terminal. However, in real world situations, this is likely not the case. One of the major benefits of contactless transactions is that cardholders do not have to take their card out of their wallet. We tested this scenario by placing the bank card in a typical leather wallet. One other contactless card is placed in the wallet, together with other (non-contactless) cards and some metal coins. The results were not very consistent (e.g., the exact position of the coins has a significant influence in the transaction time), but the typical transaction time is increased

with a factor 1.5. Small responses have a slightly smaller increase (approximately factor 1.3) and large responses have a slightly larger increase (approximately factor 1.7) in response times. The slowest card has, of course, the largest increase of transaction times. Direct transactions typically take 637 ms, but transactions with a card inside the wallet typically take 969 ms. This is a very significant finding as this makes it practically impossible to determine the expected transaction time with sufficient accuracy, as a POS terminal cannot know whether the card is placed directly to the reader or whether the card is inside a wallet.

## 7.3 POS Terminal Timings

Now that the timings of the cards in combination with an emulated POS terminal are known, naturally the question arises how these timings compare with timings from interactions between card and a real POS terminal. With the use of special hardware, it is possible to achieve very accurate timings of the communication[44]. However, we did not have such hardware, so we had to improvise to get useful timings. Unfortunately, it is impossible to know when the reader is sending a command and when it receives a response from the card without special hardware. It might be possible by opening the reader and inspecting the bus lines, but since the reader is tamper-proof this will likely break the operationality of the reader.

The user is notified when the interaction of the card and the real POS terminal is ended with a beep from the POS terminal. This could make it possible to measure timings of complete transactions, but it is difficult to measure when the interaction begins. To start a transaction, the card has to be placed in the field of the POS terminal, but the interaction can begin any time between when the card is 5 cm away from the reader and when the card is physically tapped against the reader. Because the transaction time is relatively short (typically below 500 milliseconds), it is practically impossible to measure timings even of complete interactions.

However, we developed a method to determine how the timings of a real POS terminal compare to the timings of an emulated POS terminal. In particular, we developed an application for the Nexus 7 that emulates a bank card. We call this the 'benchmark' app. One difference between timings measured with this benchmark app and timings measured with the app discussed in Section 7.2 is that the benchmark app measured the timings on the Android device while the measurements from Section 7.2 are measured by the emulated POS terminal. Another difference is that the speed of the timer of the benchmark app starts at the moment the benchmark app has fully received the first command, while the timer of the emulated POS terminal starts the moment before it sends the first command. The timer of the benchmark app stops as soon as the last response is fully sent, while the timer of the emulated POS terminal stops as soon as it has

[44] `http://cp.literature.agilent.com/litweb/pdf/5990-3443EN.pdf`

fully received the last response. As a result, the start and stop moments of the timers are not exactly the same for the benchmark app and the emulated POS terminal.

As it turns out, the timings of the terminal measured by the benchmark app are quite similar to the timings of the bank cards measured by the emulated POS terminal. The benchmark app typically shows transaction times 5 milliseconds lower than the emulated POS terminal does. However, some timings of the benchmark app are up to 10 milliseconds slower, and some are up to 15 milliseconds faster than the timings of the emulated POS terminal. This indicates that the Android app has a fluctuating performance. This conclusion is strengthened by the timings of the Nexus 7 apps in Table 7.5, where the maximum value can be up to four times the typical value and the minimum value can be down to a half of the typical value. One possible explanation for this phenomenon is that the Android operating system can give other running processes priority over our benchmark app at any time.

**Timing Constraints** We developed functionality for our benchmark app to delay certain responses with predefined delays. With this feature, we can measure the timing constraints of real POS terminals. From our experiments, it follows that there are no restrictions for the time between a command and the corresponding response for the Atos Wordline Yoximo POS terminal. Furthermore, the only restriction found is that the transaction cannot take more than 52 seconds. This is the same timeout limit as for the session time, starting at the moment the POS terminal is ready to communicate with the card. It notifies the customer when it is ready to communicate by displaying the message 'Please present card'. Because this timeout limit is over hundred times the actual typical transaction time, it is no restriction to even the most unsophisticated form of a relay attack. We can only assume this limit serves a convenience purpose and not a security purpose. This assumption is strengthened by the fact that, for example, if the terminal has waited 30 seconds before a card is presented, the communication of the transaction times out after only $52 - 30 = 22$ seconds.

## 7.4 Relay Device Timings

This section presents the timings of the Nexus 7 emulating different bank cards. These results are crucial to determine how fast the Relay Device relays the responses back to the POS terminal. If the Relay Device sends the responses to the POS terminal slower than a card does, it would be more easy to detect a relay setup for the POS terminal. If, however, the Relay Device sends the responses faster to the POS terminal than the bank cards, caching would make the total transaction time shorter and this could mean that a relay setup performs even faster than a legitimate transaction performed directly with the card.

| | | Maestro type A | Maestro type B | Visa |
|---|---|---|---|---|
| Select PPSE | min | 13 | 13 | 16 |
| | mode | 20 | 22 | 27 |
| | max | 37 | 46 | 42 |
| Select AID | min | 11 | 11 | 14 |
| | mode | 22 | 22 | 21 |
| | max | 106 | 39 | 41 |
| Get Processing Options | min | 9 | 8 | 17 |
| | mode | 14 | 14 | 27 |
| | max | 36 | 21 | 31 |
| Read Records | min | 105 | 98 | n/a |
| | mode | 156 | 132 | n/a |
| | max | 434 | 203 | n/a |
| Generate AC | min | 33 | 32 | n/a |
| | mode | 43 | 43 | n/a |
| | max | 125 | 54 | n/a |

Table 7.5: Timing results of Nexus 7 emulating different bank cards in [ms]

Table 7.5 shows the individual timings of a Nexus 7 emulating a bank card, communicating with the emulated POS terminal. As it turns out, for small messages the Nexus 7 is faster than typical bank cards. However, the Knab card is faster even for small messages. For long messages, such as the Records, the Nexus 7 is slower than all tested bank cards. Sending the AC is also significantly faster for the Nexus than for the bank cards. This is because the Nexus does not perform any cryptography, as it simply sends a message with a length and in the format of a real AC. Sending the AC for Visa cards is significantly faster than sending the AC for Maestro cards. This is because the AC message for Visa cards is significantly shorter because the Dutch Visa cards do not perform offline data authentication (see Annexes A, B, C and D).

## 7.5 Mole Device Timings

The Mole Device, a Nexus 7 model 2012, communicates significantly slower with cards than the emulated POS terminal communicates with cards. While the emulated POS terminal was appropriate for measuring card timings, it does not give a good indication of the timings of a real Mole Device. Tables 7.6 and 7.7 show

| | | ING | Triodos | ABN new | ABN old | Knab |
|---|---|---|---|---|---|---|
| Select PPSE | min | 57 | 56 | 56 | 55 | 33 |
| | mode | 59 | 57 | 56 | 57 | 33 |
| | max | 66 | 62 | 64 | 70 | 38 |
| Select AID | min | 56 | 56 | 56 | 54 | 32 |
| | mode | 60 | 56 | 65 | 55 | 34 |
| | max | 61 | 60 | 57 | 60 | 37 |
| Get Processing Options | min | 55 | 52 | 55 | 54 | 36 |
| | mode | 59 | 56 | 56 | 55 | 39 |
| | max | 61 | 58 | 58 | 57 | 39 |
| Read Records | min | 291 | 287 | 290 | 289 | 287 |
| | mode | 299 | 293 | 296 | 293 | 299 |
| | max | 306 | 306 | 317 | 308 | 318 |
| Generate AC | min | 385 | 377 | 386 | 379 | 252 |
| | mode | 390 | 380 | 394 | 380 | 255 |
| | max | 385 | 383 | 386 | 385 | 259 |

Table 7.6: Timing results of Maestro commands measured with the Mole Device in [ms]

all the individual timings for the tested bank cards with a Nexus 7 Mole Device. These two tables show that, compared to Tables 7.1, 7.2 and 7.3), the Nexus 7 acting as a Mole Device is roughly a factor two slower than an emulated POS terminal running on a fast computer. This forms an extra challenge in lowering the relay setup delay, but also indicates that there is room for improvement as Android devices become faster.

## 7.6 Network Delay

The typical network delay measured between the two Android devices is 5 milliseconds. However, we noticed that the network delay for some command and response pairs was significantly higher, typically 130 milliseconds. The affected pairs were always the `SELECT PPSE` and `GENERATE AC` and its responses. It was not immediately clear why there was a higher network delay for these commands, but after some time and experiments we figured out these two pairs share a common factor. Both pairs involve a longer period of network inactivity. The `SELECT PPSE` is the first command of the run, so although the devices were already

| | | Vodafone Sticker | Vodafone Card |
|---|---|---|---|
| Select PPSE | min | 47 | 58 |
| | mode | 47 | 58 |
| | max | 57 | 68 |
| Select AID | min | 43 | 55 |
| | mode | 43 | 55 |
| | max | 54 | 63 |
| Get Processing Options | min | 110 | 142 |
| | mode | 112 | 143 |
| | max | 113 | 145 |

Table 7.7: Timing results of Visa commands measured with the Mole Device [ms]

connected with each other, there was no network activity prior this command. The generation of the AC also takes up at least 154 milliseconds for Maestro cards and 102 milliseconds for Visa cards, during which there is no network activity. As it turns out, there is an aggressively configured battery saving feature in Android, that puts the wifi adapter to a sleep mode after around only 100 milliseconds of network inactivity. This feature is not documented in the Android documentation as far as we could find, but we did find one website discussing this behavior[45].

To circumvent this battery saving feature, the Android OS could be recompiled, but then our developed applications would no longer work on all Android devices. This decreases the number of devices which could be used for the attack, and the impact of our work would be rather limited. Instead, we developed a workaround in our applications that also circumvents this feature, by sending a heartbeat message every 80 milliseconds from Relay Device to Mole Device. Indeed the typical network delay was now lowered to 5 milliseconds for all command and response pairs.

## 7.7 Relay Setup Timings

This section presents the results of the three relay setups described in Chapter 6 with a typical Dutch bank card (from ING) and an Atos Worldline Yoximo POS terminal. In all results, we have used network timings with heartbeat messages, so that the network round trip time delay is typically 5 milliseconds.

[45] `http://stackoverflow.com/questions/13558283/`

For each relay setup we consider two relay times. These include the theoretical times determined from combining the results from Sections 7.2, 7.4, 7.5 and 7.6, and the actual timings measured when performing the relay setup. These timings can differ from each other. Performed actions can influence immediately following actions, for example, there could be a delay for switching between NFC activity and network activity. Furthermore, the Android OS can introduce additional overhead when performing multiple actions right after each other.

Sections 7.7.1, 7.7.2 and 7.7.3 present the typical timings of all relay setups measured over 100 transactions. The minimum and maximum transaction times are not included. The typical transaction time is of most interest as this is the expected transaction time. A low minimum transaction time could be interesting if the typical time is not low enough, and if there are countermeasures implemented. However, the typical relay transaction time (for the optimized setup) is lower than the highest legitimate transaction time of a direct transaction, so that even a competitive time limit would not form an obstacle. In Section 12.1.2 we give some ideas on how to improve the relay transaction speed even further. The maximum relay time is also of little interest, as this can be significantly higher than the typical transaction time but at the same time very uncommon.

### 7.7.1 Basic Relay Setup

Figure 7.8 shows the Basic Relay Setup with timings. The timings are from an ING card as this is the most available card in the Netherlands. The timings on the left side in the figure are from Table 7.6 (ING column), the timings on the right are from Table 7.5. There are eight commands and responses send over the network in total, so there is an additional 40 milliseconds network delay. The typical relay time based on the basic setup is calculated by adding the Mole Device delays, network delay and the Relay Device delay. This sums up to a typical relay time of $867ms + 40ms + 233ms = 1140ms$.

Upon testing the whole relay setup, the relay time is a little bit higher. The typical relay time of the basic setup is 1161 milliseconds. This 21 milliseconds additional delay can be caused by additional overhead in Android applications or simply because the measured timings are not consistent enough.

Figure 7.9 shows the transaction times per card of legitimate transactions, transactions performed with the Basic Relay Setup and transaction performed with the Transaction Time Optimized Relay Setup.

### 7.7.2 Transaction Time Optimized, Preloaded, Relay Setup

Figure 7.10 shows the Transaction Time Optimized Relay Setup with timings. The timings on the left are again from an ING card (see Table 7.6). The timings on the right are from Table 7.5.

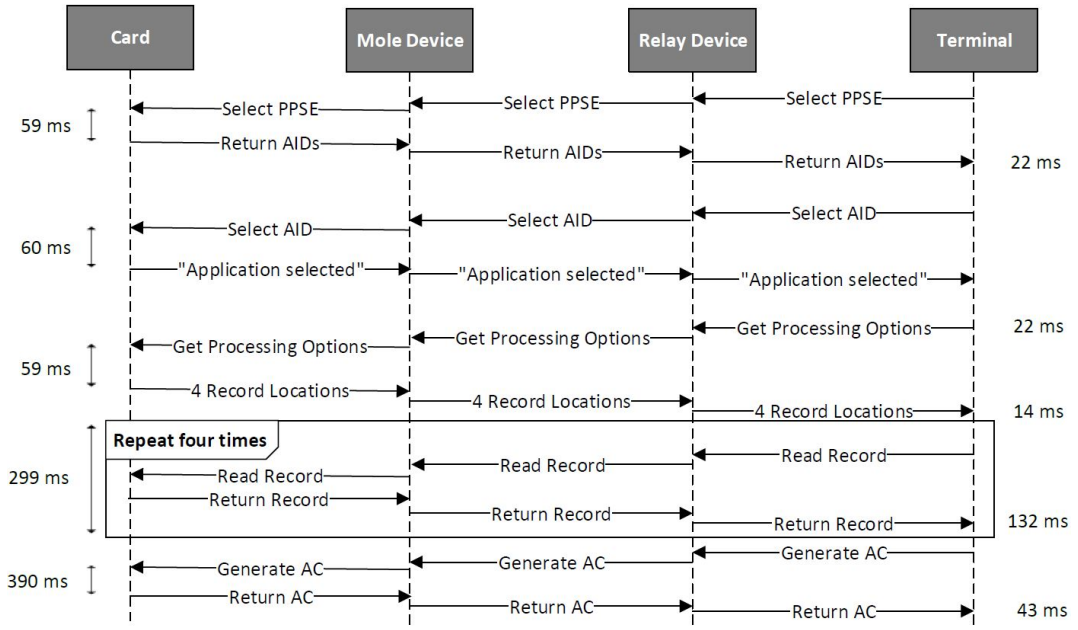


Figure 7.8: Basic Relay Setup communication diagram with typical time delays

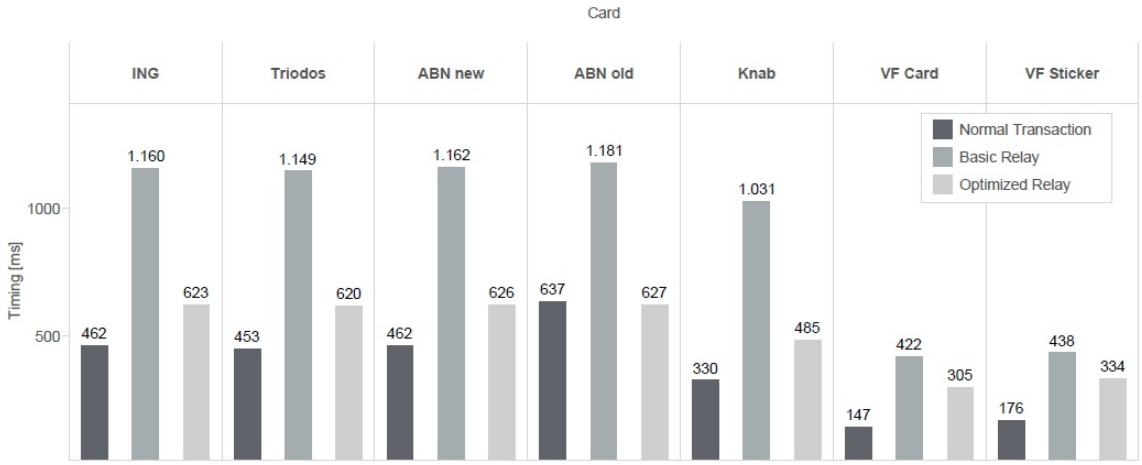


Figure 7.9: Transaction times per card for normal transactions, transactions with the Basic Relay Setup and transactions with the Transaction Time Optimized Relay Setup

With this Transaction Time Optimized Relay Setup, not all delays shown in the figure should be taken into account when calculating the total relay time. For instance, the mole delays for selecting AID and `GET PROCESSING OPTIONS` should not be taken into account, because these commands are performed simultaneously with the interaction between Relay Device and POS terminal, and are performed

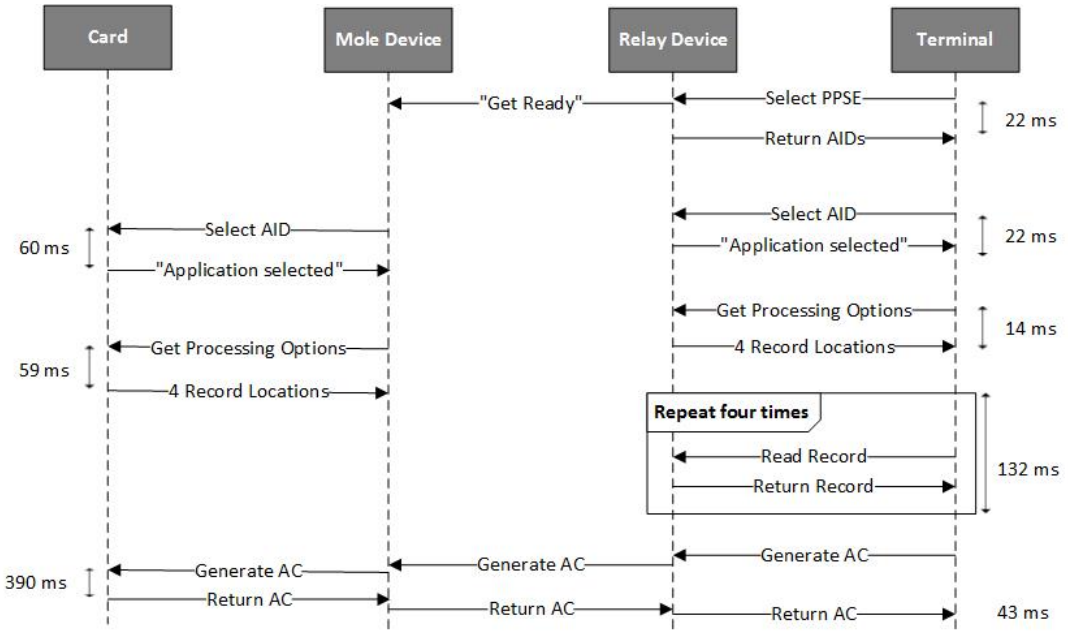


Figure 7.10: Transaction Time Optimized Relay Setup communication diagram with typical time delays

faster. The first two interactions between card and Mole Device are already finished by the time the Relay Device receives the `GENERATE AC` command from the POS terminal. The delay of the first network message should also not be included when calculating the relay time, as no device is actively waiting on this message. Only the additional delay of one command and response pair over the network needs to be considered.

The typical transaction time based on the Transaction Time Optimized Relay Setup is calculated by adding the Mole Device delay for generation of the AC, the network delay of the last two messages and the Relay Device delay. This sums up to a typical transaction time of $22 + 22 + 14 + 132 + 43 + 390 + 5 + 43 = 628$ ms (see Figure 7.10).

All Maestro card relayed transaction times are lower than the transaction time directly performed with the old ABN card. Furthermore, All Visa card relayed transaction times are below the 500 ms limit that is specified in the Visa specifications. Therefore, no possible time restriction limits exist for the Dutch card without excluding legitimate transactions performed directly with some cards.

**Slowest Card** The most remarkable result is the relayed transaction time for the slowest card (the old ABN card). The typical transaction time using the Transaction Time Optimzed Relay Setup for this card is measured at 627 ms.

The typical transaction time, performed directly with this card is 10 ms higher, at 637 ms. From Table 7.6 it follows that the slowest card is not slower than other cards when used in combination with our Mole Device. The Mole Device is obviously the limiting factor as Table 7.1 shows that with a fast reader, the slowest card is a lot slower than the other cards.

There are multiple factors that allow the relayed transaction to be faster than a direct transaction:

- The Relay Device transmits the static responses to the POS terminal 53 ms faster than the old ABN card does (see Tables 7.5 and 7.1),
- Transactions performed directly with the card suffer from the initial 100 ms overhead (see Section 7.2), while relayed transactions do not have this overhead,
- The Mole Device needs only 48 ms more for the `GENERATE AC` command and response than the card needs for a direct transaction (see Tables 7.6 and 7.1),
- The introduced delays for the relay transaction are limited to 5 ms for the network overhead and to 43 ms for the transmission of the AC by the Relay Device (see Table 7.5).

All these factors contribute to the result that a relayed transaction with the old ABN card is faster than a transaction performed directly with the same card. To the best of our knowledge, this is the first report of a relay setup that performs transactions faster than direct transactions.

### 7.7.3 Coupling Time Optimized, Preloaded, Relay Setup

For the Coupling Time Optimized Relay Setup, it is only relevant to consider the coupling time. There is a period of arbitrary length between the moment the Mole Device receives the `GENERATE AC` command and the moment the card enters the field of the Mole Device.

Figure 7.12 shows the complete protocol with timings from an ING card. Figure 7.11 shows the typical measured coupling times of all cards.

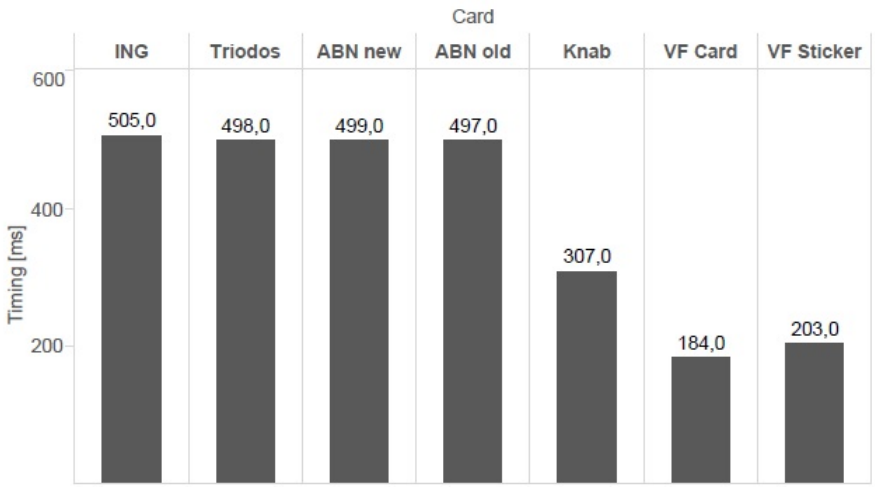


Figure 7.11: Coupling times of the Coupling Time Optimized Relay Setup

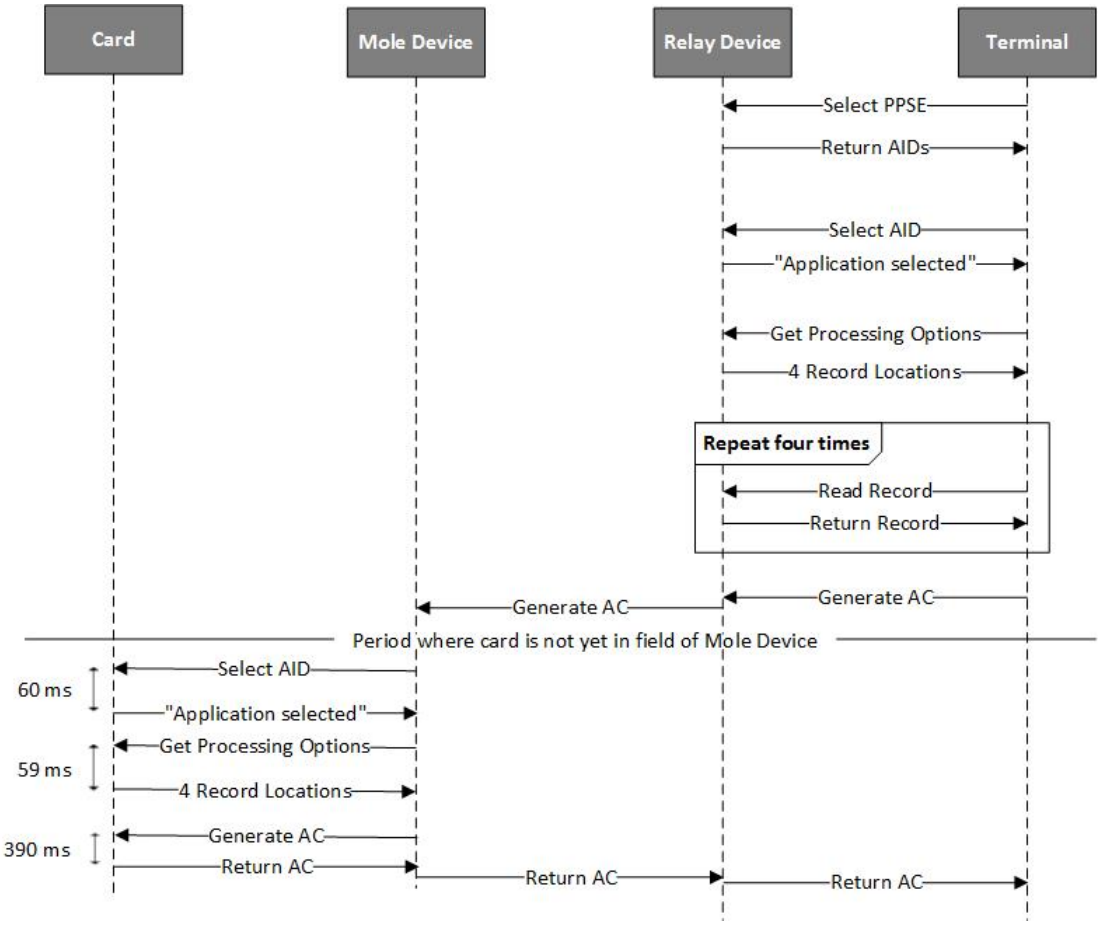


Figure 7.12: Mole Optimized Relay Setup communication diagram with Mole Device timings

# Chapter 8

# Other Vulnerabilities

This chapter describes other vulnerabilities that we discovered during development of our proof-of-concept relay attack. All these vulnerabilities were found accidentally, i.e., they presented themselves while we were working on the relay setup, and we did not search for them. Furthermore, this chapter describes the offline PIN verification enabled on some older Dutch cards, which was already identified in 2013 by students of the Radboud University Nijmegen, but which was not documented yet. All these vulnerabilities are not related to the relay attack, and are not described at the same level of detail as the relay attack. Therefore, these vulnerabilities are all bundled in this chapter.

Section 8.1 describes a discovered Denial-of-Service vulnerability of the tested POS terminal. Section 8.2 describes a misconfiguration in Vodafone cards that breaks a security requirement defined in the EMV specifications. Section 8.3 describes the vulnerability that Vodafone cards do not provide integrity of the currency in a transaction. Finally, Section 8.4 describes the vulnerability where the PIN can be verified contactlessly.

## 8.1 Denial-of-Service on Terminal

The Maestro application Type A uses four records (three of them not involved in offline data authentication and Maestro application Type B uses six records (five of them not involved in offline data authentication (see Annexes A and B). In an attempt to further improve the relay attack speed, we altered the AFL (message 7 in Figure 3.4) to trick the reader into thinking there are only two records (messages 8 and 9 in Figure 3.4) available for reading: one involved in offline data authentication and the other not. The record involved in offline data authentication remains unaltered (otherwise the CDA will fail, see Section 2.3.3). The other records, however, are combined into one large record. The ISO 7816-4 standard [31] allows these large messages and dictates that for elementary files of length greater than 256 bytes, the length tag should be preceded with tag

'82' and followed with three or four bytes indicating the length. Combining the records indeed creates a record with a length greater than 256 bytes so we must use this '82' tag. However, as soon as we sent our combined record to a Worldline Yoximo terminal, a Denial-of-Service occurs. The POS terminal does not respond anymore and displays a message 'Please wait'. After one minute the terminal starts beeping, which indicates a malfunctioning, and returns to the begin screen. However, from this point on, the terminal does not react to any button pressed or any card presented. The only way to get the terminal in an operating state again is to cut the power off and reboot it again. This Denial-of-Service could be the result of a buffer overflow.

The terminal is clearly not supposed to crash when it is presented a message that is longer than expected. The message triggering the crash is a correctly created response according to the appropriate ISO standard, nonetheless, the terminal crashes as apparently it cannot handle such long messages. Due to these circumstances, this Denial-of-Service attack presumably is the result of a buffer overflow. If this is indeed the result of a buffer overflow, then it might mean that attackers can exploit this even further and gain full access to the POS terminal.

The Betaalvereniging Nederland[46] (The Dutch Payment Association) offered us the possibility to test our Denial-of-Service exploit on other POS terminals. As it turns, another model of Atos Wordline was vulnerable to this attack (it crashed), while their Verifone POS terminal accepted the long record as a valid record.

This bug is likely somewhere in the protocol stack that is also used for contact interface. However, this Denial-of-Service can have severe consequences especially for the contactless interface. When users have multiple card emulating applications on their device (e.g., a payment app, a public transport app, a library app), it is not unlikely this bug will get triggered when an incorrect app is accidentally selected by the user.

## 8.2 Re-use of Secret Symmetric Keys in Multiple Applications

We found out that the Vodafone card has two applications per interface (i.e., two applications on the contact interface and two applications on the contactless interface). Additionally, the Vodafone sticker has two applications on the contactless interface. The applications show the exact same behavior. The only difference that we could identify was the different AID and their priority setting. Because the ATC is not shared between applications, we were able to perform a

[46] `http://www.betaalvereniging.nl/`

transaction twice with the same ATC value. This way, we can perform a transaction twice where the ATCs have the same value.

The AC generation process is discussed in Section 2.3.6. Briefly reiterated, the transaction key is different for each transaction and is derived from the secret symmetric key from the card and the current ATC value.

Because one card has two independent ATCs, we can test whether the applications use different secret symmetric keys for AC generation. When the same transaction data (amount, UN, etc.) are presented to different applications, and when the ATC is the same for the applications, then the AC generated is exactly the same. This can only mean that the applications use the same secret symmetric key. The card uses 3DES or AES, which both are strong enough, so it seems this vulnerability is not directly exploitable. However, the EMV specifications clearly state that the secret symmetric key used for AC generation must be unique [10, Section 8.1.2].

We included two different traces of the two applications found on a Vodafone card that show two identical ACs in Annexes C and D.

Because the applications on an interface behave in the same way, there does not seem to be a technical advantage. Furthermore, when the card is presented to ATMs, the cardholder is asked to make a choice between the V PAY product and the V PAY product (see Figure 8.1). This of course is not very convenient and a typical cardholder will not know what to choose. However, it could be that some POS terminals only support one application and that other POS terminals only support the other application, and that that is the reason why there are two applications on the cards.

## 8.3 Weak Generation of Cryptogram

Because we can generate two identical ACs for each ATC with Vodafone cards (see Section 8.2), we can also check over which variables the AC is computed. When all variables are kept constant except for one, an attacker can learn that this specific variable is not used for the generation when the generated ACs are identical. If the ACs are not identical, the ACs are indeed computed over that particular variable (among others). The EMV specification specifies a recommended minimum set of data elements for AC generation [10, Section 8.1.1]. This set includes the following terminal provided variables:

- Amount, Authorised,
- Amount, Other,
- Terminal Country Code,
- Terminal Verification Results,

Figure 8.1: ATM asking to select one of the applications found on Vodafone cards

- Transaction Currency Code,
- Transaction Date,
- Transaction Type,
- Unpredictable Number.

Our results show that from this recommended minimum set, only the Authorized Amount and Unpredictable Number are used for generation of the AC. It is rather surprising that, for example, the currency of the transaction is not included in the AC, while it is requested by the application. As a result, the issuer can only verify the amount and the UN, and not the currency of the transaction. Although it is only *recommended* to include the currency, we think this is rather important as the amount together with currency determine the actual value of the transaction.

## 8.4 Offline PIN Verification

The first batch of ING Bank and ABN Amro cards allow the PIN to be verified contactlessly. Furthermore, the PIN retry counter can be read contactlessly. This

was identified by students at the Radboud University in 2013, and has been fixed in later versions. However, this vulnerability has not (yet) been documented. These cards are, as far as we know, not recalled and thus still in use in the Netherlands.

# Chapter 9

# Attack Scenarios

This chapter describes the potential attack scenarios exploiting the identified vulnerabilities. Section 9.1 describes attacks where attackers can pay with the card of a victim that is still inside the wallet or pocket of the victim. Section 9.2 describes scenarios how attackers can exploit the found Denial-of-Service attack. Section 9.3 describes an attack scenario where attackers steal money by using counterfeit POS terminals. Section 9.4 describes a new type of fraud, where malware for Android devices can be used to steal money from a contactless card. Section 9.5 introduces a new form of fraud where cards are used inside an envelop when they are sent from the bank to customers.

## 9.1 Pickpocketing

One of the most realistic scenarios for this attack is that an attacker and his accomplice work together to pickpocket the victim. The accomplice goes to a store with a contactless self-scan terminal, as found for example at the Albert Heijn To Go. At self-scan terminals, the timings are not so crucial as there is no merchant that might get suspicious. In the mean time, the attacker chooses his victim. He needs to get close enough, for example by standing behind him in the queue. He can now try all pockets for presence of a bank card. Another method is that he simply waits until the victim pays so that he learns where the bank card is, and then later bumps into him, placing the Android device close to where the victim's bank card is. The attacker needs an accomplice for this scenario, otherwise he would have to hold the Relay Device in one hand to the POS terminal, and the Mole Device next to the bank card of the victim in his other hand. This is rather difficult to manage.

### 9.1.1 Without PIN Knowledge

Paying amounts less than €25 is possible with only one-factor authentication: the customer needs to prove possession (or better: access) to the card. With our proof-of-concept relay attack, we have shown that it is possible to fake possession of the bank card by holding one Android device close to the card of a victim and one Android device close to the POS terminal. The bank card does not have to be taken out of the wallet, but the device can communicate contactlessly through clothes and bags.

**Equipment** There is no additional equipment necessary besides the two Android devices and relay software. The devices can be linked together with wifi-direct without Internet access or a network router. The distance between Android devices can be easily increased because most Android phones also have access to 3G and 4G networks.

**Attackers** The most realistic profile of such attackers is that they are script-kiddies. There is no significant financial gain, but of course it can be excited to try this in real life. The profits are too limited to attract gangs or career criminals, and we did not identify a solid business case for criminals.

**Likelihood** An EMV relay app can (and most likely will) appear in Google's Play Store. In combination with the low equipment and knowledge requirements (only two standard Android devices of which one runs Android 4.4 and no additional knowledge) probably makes this one of the most likely scenarios.

### 9.1.2 With PIN Knowledge

For contactless transactions up to €2,500 or €5,000 (depending on the bank), two-factor authentication is performed. Not only needs the customer to prove possession or access to the bank card, he also needs to enter the PIN. Possession can be faked again with the relay attack, but we have not identified a way to bypass PIN verification. This means that the attacker must have a way to learn the PIN code in order to make transactions with amounts above €25. We have identified two realistic scenarios in which an attacker can learn the PIN, which are further discussed in this section.

This can be done with the use of a camera which is mounted close to the POS terminal. Alternatively, an attacker can attempt to learn the PIN by looking over the shoulder of the victim, when he enters his PIN.

[47] Picture from `http://observers.france24.com/content/20100208-bank-information-fraud-rigged-cash-points-theft-usa-romania-security`

[48] Picture from Seattle Police Department

Figure 9.1: An improvised camera solution used for recording PINs (Source[47])

Placing small, practically invisible cameras, aimed at the PIN pad of ATMs is well within the expertise of motivated attackers. Figure 9.1 shows a picture of an ATM where the camera is hidden and a picture where the camera is exposed. This particular example shows an improvised POS terminal camera (a simple

Figure 9.2: A very small camera used for recording PINs (Source[48])

cell phone with extra batteries attached), but more sophisticated cameras are also discovered, for example as shown in Figure 9.2.

In the case of a relay attack, an attacker probably wants to wait close to the ATM so see where the victim puts his bank card. If the attacker realizes a live stream from the camera, he can wait around close by and immediately follow his victim to perform the relay attack once he has learned the PIN.

There are reports[49] of criminals looking over the shoulder of victims to learn their PIN. With traditional contact transactions, the criminals also need to steal or otherwise get the bank card before he can make a transaction. With contactless cards, however, possession is not necessary as a relay attack can be performed.

The attackers will likely buy goods that are easy to monetize, such as prepaid cards (iTunes credit, call credit, etc.). If contactless ATMs are encountered in the future, this is of course their preferred point of cash out, as selling stolen goods always comes with a fee for the criminals.

**Capturing PIN with Infrared Camera** Recently, a video[50] has been published about an infrared camera for the iPhone, called the FLIR ONE[51]. The presenter claims that the camera can take pictures of a POS terminal and capture the previously entered PIN. Figure 9.3 shows a still from the video, clearly showing the entered PIN (1-2-3-4-5).

Figure 9.3 shows a warm environment and a cool POS terminal. This indicates that the POS terminal is significantly colder than the environment (maybe because it came out of a refrigerator). Furthermore, the evenly distributed heat

[49] `http://www.politie.nl/nieuws/2014/februari/15/01-diefstal-geld-na-afkijken-pincode-pinpas.html`

[50] `https://www.youtube.com/watch?v=8Vc-69M-UWk`

[51] `http://www.flir.com/flirone/`

Figure 9.3: Still from video claiming PINs can be captured with FLIR ONE

of the POS terminal indicates that it is not turned on (the screen of the POS terminal is also conveniently not shown).

We have purchased the FLIR ONE to verify the possibility to capture PINs. We have identified two serious limitations of this method. Firstly, POS terminals that are operational, typically show a very irregular heat pattern. This makes it almost impossible to capture areas that are touched. Secondly, capturing touches on metal buttons does not work because metal is too reflective.

Figure 9.4 shows two pictures we have taken with the FLIR ONE right after a PIN was entered. The left picture is taken right after we entered 1-2-3-4-5, the right picture is taken right after we entered 6-7-8-9-0. There are no hints in the pictures to reveal even a small portion of the PIN we entered.

Furthermore, this attack might only work if all of the following conditions are met:

- The buttons of the POS terminal are not metal,
- The (untouched) POS terminal shows a regular heat pattern,
- The temperature difference between POS terminal and fingers is significant enough.

However, these are not normal conditions, making it extremely unlikely attackers will be able to use this in practice. We tested this on a random reader

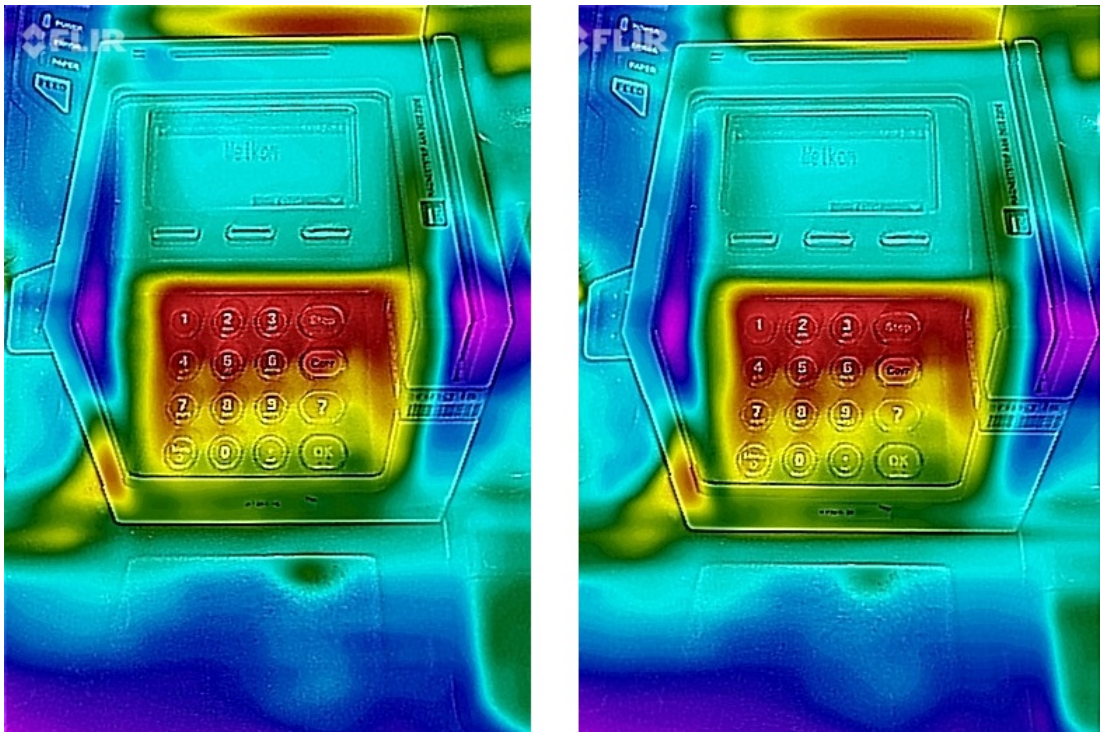

Figure 9.4: Self made pictures of POS terminal with FLIR ONE. Left we entered 1-2-3-4-5 as PIN and right we entered 6-7-8-9-0 as PIN

(shown in Figure 5.4). The random reader indeed does not have metal buttons and shows a regular heat pattern (as it is only turned on right after insertion of the card). However, the temperature difference between the random reader (approx. 20 °C) and fingers (approx. 35 °C) was not enough to hold the button presses visible for more than 5 seconds.

## 9.2 Denial-of-Service

We have identified two realistic scenarios to exploit the vulnerability which occurs when long messages are sent to the POS terminal. We explain them in Sections 9.2.1 and 9.2.2.

### 9.2.1 Denial-of-Service on Terminals

The discovered vulnerability leads to a non functional POS terminal. Because the power needs to be turned off and on for the terminal to become functional again, this can be a huge inconvenience for the store. Cashiers generally do not restart terminals themselves, so technical support is probably called.

We have identified three attacker profiles that potentially gain benefit from a Denial-of-Service on POS terminals. Their profiles and motives are discussed in the remainder of this subsection.

**Competitors** Competitive store owners benefit if the customers cannot pay at the victim's store: The customers are likely to go to a store that does have functional POS terminals. Especially on high revenue days, for example the day before Christmas, this can lead to significant losses for the victim store and profit for the competitive store.

**Activists** Activists sometimes perform Denial-of-Service attacks on stores or shops they do not agree with. For example, environmental activists sometimes do this at gas stations, by putting locks on gasoline guns or by blocking the entrance. A more convenient and easier way for them would be to disable the POS terminals with the identified Denial-of-Service attack. Especially with self service pumps this can be done quite anonymously, without interacting with the merchant.

**Scriptkiddies** Scriptkiddies who enjoy conducting mischief, might want to disable some POS terminals for 'fun'. Furthermore, they could gain financial benefit if they perform this at the Dutch supermarket Jumbo, which has the policy that the fourth waiting person in the queue gets all the groceries for free[52]. With this attack, it is easy to block all POS terminals but one, and create an artificial long queue.

### 9.2.2 Exploiting a Potential Buffer Overflow

It is not entirely clear what the root cause of the Denial-of-Service attack is. The most plausible explanation, we think, is that a buffer overflow occurs when a message is sent that is longer than the 256 bytes that the POS terminal expects.

If this is indeed the case, The most serious threat is a situation where an attacker can execute his unsigned code on the terminal. This way, there are no guarantees anymore on the integrity of the amounts, the confidentiality of the PIN or the availability of the POS terminal. However, an attacker would first have to invest a significant amount of resources in exploiting the Denial-of-Service and figure out whether it occurs because of a buffer overflow. The firmware is not publicly available making it very time consuming to successfully exploit this bug. Nonetheless, executing own code on a POS terminal can generate a significant amount of money for the attacker.

[52] `http://www.jumbosupermarkten.nl/Homepage/Service/Jumbos-7-zekerheden/Vlot-winkelen/`

## 9.3 Counterfeit POS Terminals

Criminals can create counterfeit POS terminals, which performs transactions of €25 to their account each time a card is presented. This counterfeit terminal can be attached on top of a legitimate terminal (most likely a self service terminal), and block signals to the legitimate reader. The legitimate terminal can also be disabled completely (for example with a Denial-of-Service attack as described in Section 9.2.1), and a counterfeit POS terminal can be placed next to it, forcing all customers to use this counterfeit terminal. While this might be quickly discovered during a normal working day, the criminals can receive a significant amount of payments of €25 throughout the weekend when they perform one of these attacks, for example, on a Friday night in an unmanned parking garage.

## 9.4 Malware on Mobile Phone

A completely different attack scenario is one where attackers distribute malware on mobile phones. Many people have a mobile phone cover where they also store there bank card (see Figure 9.5 for an example). This malware can be used to guess PIN codes on older cards. Every time the victim uses his card for a legitimate transaction with PIN, the malware can verify whether the PIN counter is set to three again, and guess another PIN code. Once the malware has found the correct PIN code, it can make a callback to a server of the criminal, indicating that it is now ready to perform relay attacks. The criminal, who already received the PIN from the malware, can now make payments up to €2,500 or €5,000 per day (depending on the bank). If he does this around midnight, he can easily make €5,000 or €10,000 in a short period of time. The malware can be distributed for example in cloned apps available in unofficial app stores[53].

Furthermore, even if the bank card in the phone cover does not support contactless PIN verification, the malware can be used to do two relayed payments of €25 each time after the victim uses its card for transactions with PIN.

## 9.5 Envelope Fraud

Envelope Fraud is a term we introduce for the scenarios where a relay attack is performed with a card that is sent from an issuer to the customer in an envelope. Most banks have good countermeasures against this, except for Knab bank. A postman or a neighbor can easily gain access to the envelope with bank card in it to quickly perform a relay attack (for example, as described in Section 9.1). If this happens somewhere inside the mail sorting process, the scale of the fraud

[53] http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/madware_and_malware_analysis.pdf

Figure 9.5: An example of a phone cover that also stores bank cards

can be increased significantly. The most dangerous part of this attack is that the transaction is performed without opening the envelope. The rightful owner of the card cannot even know his card was involved in a transaction.

PIN codes are also sent with the mail, so an attacker (e.g., a neighbor or postman) could steal and open the post containing the PIN. This significantly increases the gain of the attack. With contact cards, an attacker could of course also steal both the envelope containing the PIN and the envelope containing the bank card. However, the likelihood of stealing both envelopes is lower than the likelihood of stealing only one envelope (containing the PIN) and tapping the other envelope (containing the contactless card) against an Android device. Furthermore, stealing both envelopes is likely to raise more suspicion than a single 'missing' (stolen) envelope.

## 9.6 Attacks on PIN Verification

This section describes two identified vulnerabilities as a result of the offline verification of the PIN being available on some older bank cards (see Section 8.4). The

first vulnerability is identified by us and the second one was already identified by the authors of [7], and is also applicable to the Dutch cards.

**Denial-of-Service on Bank Cards** A cardholder can make three incorrect guesses sequentially before the card is blocked and cannot perform transactions anymore. An attacker can easily make three incorrect guesses to block the card. This procedure takes approximately 500 milliseconds.

**Free PIN Code Guesses** Because the retry counter is reset with every legitimate transaction where the PIN is entered, an attacker can get a practically unbounded number of guesses for the PIN code, if he has multiple occasions were he can interact with the card [7]. He has to make sure there are enough PIN tries left, so that he does not block the card. Every time the victim uses his PIN for a legitimate transaction, the attacker gets another two extra tries for guesses.

# Chapter 10

# Countermeasures and Recommendations

This chapter gives an overview of existing and proposed countermeasures. Section 10.1 gives an overview of the existing countermeasures. Section 10.2 describes the proposed countermeasures. Finally, Section 10.3 discusses the potential effectiveness and the consequences of all these countermeasures.

## 10.1 Existing Countermeasures

This section describes the existing countermeasures that we encountered. Section 10.1.1 describes the existing limits on the amounts for contactless transactions and their role as countermeasure. Section 10.1.2 describes the existing countermeasures against envelop fraud that some banks use.

### 10.1.1 Amount Limit of Contactless Without PIN

One of the most effective countermeasures against large scale relay attacks is the limit for PIN-less transactions. The €50 limit for consecutive transactions without PIN ensures there is no business case for pickpocketing criminals. However, we do not think the €25 limit for a single PIN-less transaction is very useful, because if an attacker can perform one relayed transaction, it is very likely he can also perform another.

### 10.1.2 Countermeasures to Envelope Fraud

Some banks have implemented countermeasures against envelope fraud (as described in Section 9.5). In particular, we encountered cards that cannot generate valid ACs contactlessly before a contact transaction (with PIN) was performed. Other banks have the restriction that the customer first must login to a website

where the card must be activated. The first countermeasure is implemented in the application on the card, is more user friendly but only possible if the card also has a contact interface. The second encountered countermeasure requires additional steps from the customer and thus it is less friendly, but also works on cards without contact interface. However, Knab bank does not have such countermeasures (see Section 9.5). As a result, we were able to perform a payment with a Knab card through the envelope.

## 10.2 Proposed Countermeasures

This section describes the proposed countermeasures. Section 10.2.1 describes the proposed countermeasures against relay attacks. Section 10.2.2 describes countermeasures to prevent shoulder surfing attacks. Section 10.2.3 describes our proposed countermeasures against shoulder surfing attacks. Section 10.2.4 discusses countermeasures against relay attacks performed with malware and phone covers that hold payment cards.

### 10.2.1 Countermeasures on Relay Attacks

This section describes two newly proposed types of countermeasures against relay attacks. Section 10.2.1.1 describes timing restriction based countermeasures. Section 10.2.1.2 describes an emulation detecting countermeasure.

#### 10.2.1.1 Timing Restriction

This section describes two timing restriction based countermeasures. Section 10.2.1.1 describes timing restrictions by the POS terminal, and Section 10.2.1.1 describes timing restrictions in the back end.

**POS Terminal Timing Restriction** In Chapter 6 we already noted that timings restrictions, implemented at the POS terminal, are the most obvious countermeasure to detect and reject relay attacks in general. Restrictions can be placed on the response time of static responses, dynamic responses and on entire transactions.

Timing restrictions on static responses are not effective against our relay attack, because static responses are sent faster by the Relay Device than directly by a bank card. Timing restrictions on dynamic responses and entire transactions cannot be lower than the worst case delay of a legitimate response or transaction, because otherwise legitimate transactions could be denied. As stated in Section 7.2, there is a significant difference between timing of different cards. Furthermore, there is a significant timing difference between a card when it is

inside a wallet and when it is hold directly to the reader. Therefore, timing restrictions on dynamic responses or entire transactions will not be effective for the vast majority of the Dutch cards. However, timing restrictions do make it harder for criminals and relay attacks need to be more sophisticated. Furthermore, they might prevent relay attacks performed on the slower cards in the Netherlands.

**Back End Timing Restriction** One of the problems with the discussed solutions in Section 10.2.1.1 is that the POS terminal does not know how fast the card *should* be that it is communicating with. However, the bank does know the characteristics of each card (or at least, could know). Therefore, if the POS terminal would send the transaction (or dynamic response) times to the bank together with the transaction details (such as the AC), then the bank can make an informed decision about the risk of a relay setup. Such a modification requires different software on the terminal (transaction times currently are not sent to the bank) and the back end system needs additional checks. No new hardware has to be issued, however, since the cards do not have to be altered. However, this proposed countermeasure does not fix the problem that cards are 1.5 times slower when they are inside a wallet. The POS terminal does not know when the card is inside the wallet, and therefore the bank cannot know either, reducing the potential effectiveness of this countermeasure.

#### 10.2.1.2 Card Emulation Detection

Android devices in HCE mode have many different characteristics compared to contactless cards. Android devices send signals with multiple adapters (e.g., GSM, wifi) which could be detected by the POS terminal. Furthermore, different NFC chips in Android devices can be detected on a lower level than the application level, distinguishing them from real cards. This way, the terminal can detect relay attacks without altering the communication between terminal and card or terminal and bank. Therefore, this might be an attractive solution, since a check, for example, on the ATR of the presented device is easy to implement and fast. The ATR of a smart card and an Android device differ significantly and it is not hard to distinguish between these two. However, as relay attacks mature, attackers will become able to configure their Android device so that these different characteristics are no longer detectable. Rooted devices can be programmed to fake ATRs as well as temporarily disable the broadcasting adapters, but it will definitely slow attackers down.

### 10.2.2 Reducing Maximum Amount Contactless With PIN

The maximum amount that can be stolen with a relay attack when the PIN is known, is limited to €2,500 or €5000. This amount, however, is not advertised by banks and this information is not available on any bank website. We contacted

Figure 10.1: Transaction receipt of contactless payment of €60 with PIN

the customer supports of the four banks issuing contactless cards, and we got four different answers. Two out of four customer supports claimed contactless transaction can never be performed with PIN code. Another customer support claimed that the limit for contactless transactions with PIN is €50 and the last customer support claimed that the limit for contactless transactions with PIN is the same as the limit for contact transactions.

We proved that it is possible to perform contactless transactions with PIN for amounts higher than €50 with at least two different issuers (see Figure 10.1). Inquiry taught us that the contactless limit indeed is equal to the contact limit (typically €2,500 or €5,000, depending on the bank). The damage of a relay attack when the PIN is known is thus limited to €2,500 or €5,000.

### 10.2.3 Preventing Shoulder Surfing

When the PIN is known, pickpocket criminals can perform relay transactions up to €5,000 (see Section 10.2.2). Effectively, the PIN is worth up to €5,000 to criminals. Therefore, it is rather silly that the PIN is also used for every contact transaction, no matter how small the maount.

If cardholders would not have to enter their PIN as much, attackers have less opportunities to perform shoulder surfing attacks. A lost or stolen card already means that the finder or thief can perform contactless transactions up to €50 without PIN. So if the contact interface would use the same (shared) limit of €50 for PIN-less transactions, this would not decrease the overall security, as attackers can still only steal €50 without PIN.

### 10.2.4 Android Malware Prevention

Malware on mobile devices (as described in Section 9.4) is really difficult to prevent and manufacturers and developers do not seem to be able to effectively apply countermeasures against this. Furthermore, it can not be expected from the issuers of EMVCo that they prevent this type of fraud. However, cardholders can protect themselves against this attack by not storing their payment card in a phone cover.

## 10.3 Recommendations

This section gives an overview of the previously discussed countermeasures. Furthermore, we give our recommendations on how to detect and defend against relay attacks and how to mitigate the risks.

**Contactless PIN Amount Limit** We think it is not necessary to be able to perform transactions up to €5,000 contactlessly. With such transactions, cardholders would probably not mind taking out their card of their wallet and inserting it into a POS terminal. At least as long as relay attacks are still possible, we think a significantly lower contactless PIN limit would be more appropriate. To provide a better user experience, banks could also let their customers decide on the limit (e.g. on the banking site).

**Envelope Fraud** Envelope fraud can be easily prevented by one of the two existing countermeasures. Our recommendation is to enable contactless transactions only after a contact transaction with PIN has been made. If the card does not have a contact interface, activation via Internet is also an effective countermeasure, although slightly less user friendly.

**Timing Restrictions** Restricting the response time for static responses has no use, as a Relay Device can cache these responses. Restricting the response time for dynamic responses or complete transactions is more effective but does not prevent relay attacks completely, as we have presented a relay attack that is actually faster than a legitimate transaction. It makes it slightly more difficult for an attacker to perform a relay attack in practice, because it reduces the attack window that the attacker has to pickpocket his victim while the accomplice holds his Android device to the POS terminal. The attack window with the tested POS terminal is 52 seconds. If the timing restriction is lowered to, let's say, two seconds, it is loose enough to accept the vast majority of cards with a very high probability, as cards with a legitimate transaction time of over two seconds are probably not very common, if they exist at all. Furthermore, it gives the attacker and accomplice an attack window of only two seconds to simultaneously present the Relay Device to the POS terminal and the Mole Device to the card, which is significantly more difficult to realize than with an attack window of 52 seconds.

Timing restrictions can be made stricter and different for each card if they are determined by the bank. The bank can know what the expected time of a transaction with the card is. This is a timing restriction that must be placed in the back-end, and all POS terminals must be changed so that they send the transaction time to the bank. However, the bank cannot know whether the card is inside a wallet or presented directly to the POS terminal. Therefore, transaction times up to a factor 1.5 slower than the expected transaction time still need to be accepted. This significantly decreases the effectiveness of this countermeasure as attackers can use this variance to perform relay attacks.

The proposed countermeasures do not completely prevent relay attacks, and all POS terminals would need different software. Adding timing restrictions to only all Dutch POS terminals does not suffice, because as long as there is one POS terminal somewhere in the world that does not send the transaction times, and the bank accepts these transactions, then that POS terminal can still be used by attackers. Therefore, the effectiveness is limited and the financial impact is high.

**Card Emulation Detection** A check on the ATR of the device is simple and cheap to realize, and effective at least on the short term. In particular, changing the ATR is not possible in Android, so the relay attack cannot be performed with standard Android devices. However, it is not a long term solution as attackers will probably succeed in creating (practically) undetectable card emulators by modifying Android. In addition, this would mean that all POS terminals need to be adjusted, which is not very realistic.

**Contact Amount Limit** We think the payment system would actually become more secure if people would not have to use their PIN so often. Lost and stolen

cards already allow €50 to be stolen from the card contactlessly, so we see no reason that the PIN should be entered for small transactions performed with the contact interface. While allowing contact transactions to be PIN-less (up to €50) dramatically decreases the number of opportunities for a shoulder surf attack, it might also decrease the security perception of the users. Therefore, we propose to let contact transactions share the €50 limit that contactless transaction use, but also to let users decide (e.g. on the banking site) to lower one or both of these limits.

**Malware on Mobile Phone** There is not much banks can do to prevent malware from performing relay attacks. However, they could advise their customers to not keep their payment card in a cover next to a NFC-capable mobile device. This countermeasure would be very effective as is it very unlikely that malware on a device can perform a relay attack on a contactless card if this card is not stored in a cover close to the device.

# Chapter 11

# Discussion

In 5 months of research work, multiple vulnerabilities in the Dutch EMV Contactless system were identified. With minimal resources, we developed applications for relay attacks with only two standard and widely available Android devices. No countermeasures were identified to prevent or detect relay setups. Furthermore, we accidentally identified an additional vulnerability in the tested POS terminal, two misconfigurations in Visa cards that classify as vulnerabilities and one Maestro card that did not implement any protection against envelope fraud.

Section 11.1 discusses the EMV Contactless specifications and Section 11.2 discusses the implementations of EMV Contactless. Section 11.3 discusses the certification process of cards and POS terminals. Section 11.4 discusses the limits or our proof-of-concept. Section 11.5 describes the timeline of our research. Finally, Section 11.6 describes what the media has reported about EMV during our reserach.

## 11.1 EMV Contactless Specifications

The EMV Contactless specifications are described with complex formulations, technical details scattered throughout the books and multiple ambiguities. Basic security requirements are absent (such as relay detection), and the focus is placed too much on operability, backwards compatibility and flexibility.

While MasterCard and Visa use the same specifications for the EMV Contact standard, they both developed their own specifications for EMV Contactless. The result is a very flexible standard for MasterCard on the one hand, and fast transactions for Visa on the other hand. However, all parties would benefit if they join forces to create one flexible, fast and future proof standard. Given the many vulnerabilities that have been identified over the years, especially for legacy modes (Mag-Stripe Mode, unencrypted PIN verification, fallback attacks, CVC codes, etc.), we think all parties should seriously consider dropping support for all legacy functionality.

EMVCo does not claim any guaranteed security for their EMV specifications. In some sections, they give recommendations concerning security but issuers are free to ignore those and create their own implementation. The result is that issuers interpret the specifications in different ways, and multiple implementations are created. Some issuers appear to work together for their applications (e.g., ING, ABN and Triodos have the same applications), but other issuers (e.g., Knab) have their own application. It is positive that banks work together on this level. A clear example of this is that the collaboration of ING, ABN and Triodos recognized the dangers of envelope fraud and implemented an effective countermeasure, but Knab has no countermeasures concerning this type of fraud. Furthermore, Vodafone ignores the recommended minimum set of information for the cryptogram generation, and indeed creates insecure cryptograms instead.

Security guidance in the EMV Contactless specification is heavily underspecified. Security features are optional and the specification is complex and too long. Payment systems would benefit if they use a specification that has security as a design feature. However, EMV is integrated in basically all payment systems worldwide, and it is not very realistic to expect that EMV will be replaced by a new or completely changed standard anytime soon. Furthermore, legacy modes can be dangerous for the security, but they are of course crucial for compatibility between banks and countries. In particular, there are many countries that do not have Internet widely available or still heavily depend on magnetic stripe technology. Therefore, these countries are dependent for legacy modes and removing them would break the compatibility of EMV.

## 11.2 EMV Contactless Implementations

Chapter 3 shows that there are many options and modes available according to the EMV Contactless specifications. Many of them (e.g., SDA, DDA, MagStripe Mode, unencrypted PIN verification) cannot be considered secure. The EMV specifications do not provide sufficient security requirements or guidance, and the implementers are responsible for choosing the options and modes they find appropriate. We have seen three different implementations on Dutch bank cards: two different Maestro applications and one Visa application. We did not find any vulnerability in the Maestro application used by the major banks, but the other Maestro application (used by a niche bank) is lacking envelope fraud prevention. Furthermore, the Visa application also shows some issues concerning key management. In particular, the application breaks the security requirement on the uniqueness of the secret symmetric key. Furthermore, this allowed us to examine the generation of the cryptogram. As it turns out, the implementers decided to not include the currency in the cryptogram. This really is rather a dangerous design choice, as an amount without a currency can have significantly differing values. However, re-using the secret symmetric key and excluding the

currency from cryptograms cannot be practically exploited in the Netherlands.

Issuers would benefit greatly if all would join forces and create one robust and secure implementation, rather than trying to do it all on their own. We understand different countries might have different needs, but multiple implementations in one small country such as the Netherlands seems unnecessary.

## 11.3 Certification Process

It is not clear how the certification process of applications and the certification process of POS terminals is defined or what exactly is tested. However, it seems clear to us that both are suboptimal.

**POS Terminal Certification** The POS terminal crashed when presented with a longer message than it expected. In our opinion, input validation is one of the first things that should be checked in a certification process. This bug reveals the absence of a basic form of input validation, and this suggests that many more bugs would be revealed if the POS terminal was subject to fuzz testing (i.e., provide the POS terminal with large amounts of unpredicted, incorrect or random input). Furthermore, this really can become a big issue as customers will have multiple HCE applications on their smart phone in the near future, and they might select an 'incorrect' one. The tested POS terminals also did not have any protection against a relay attack.

**Card Certification** The contactless cards have been on the Dutch market for one year now, and already many problems have been identified. It began with the discovery of the possibility to do perform PIN verification contactlessly in 2013. We identified a Visa card with a misconfiguration regarding the secret key, and a sloppiness in the distribution of Knab cards. However, the sloppiness in the distribution of the Knab cards might be out of the scope of card certification. Nonetheless, all these findings suggest that the issued cards are not subjected to a strict test or certification process, since we found them while where we actually trying to perform a relay attack, and not while we were searching for vulnerabilities. Therefore, it is our expectation that more bugs, errors or other vulnerabilities will be found when the other Dutch banks start to issue contactless cards, if they do not all use a thoroughly tested application.

## 11.4 Limitations

This section discusses the limitation of our relay attack. Section 11.4.1 discusses the limitations of the distance between the victim's card and the Mole Device.

Section 11.4.2 describes the limitation of the common situation where multiple contactless cards are placed close to each other (e.g., in a wallet).

### 11.4.1 Reading Distance RFID Signals

The maximum possible distance between the victim's card and the Mole Device is of big importance for the practically of an attack. Four years ago, the maximum range of an Adroid device with NFC was approximately 1 cm. Now, in 2014, the range is extended to (depending on the device) approximately 5 cm. 5 cm is enough for the relay attack to be practical, as we have shown that the victim's card can be read if it is inside a wallet that is inside a pocket of the victim. However, the Mole Device needs to be close to the pocket in order to perform the relay attack. If the range would be further extended, let's say, to 10 cm, the attack becomes significantly more practical and attackers have a higher success rate because the placing of the mole does not have to be very accurate. If the range would be even further extended to, let's say, 25 cm (as already achieved in [23]), then the attack could be performed on a large scale by automatically pickpocketing from all victims that walk, stand or sit close by a Mole Device placed in a crowded location. Eventually, the maximum distance could be even further improved up to 55 cm according to [32].

### 11.4.2 Multiple RFID Cards in One Wallet

As contactless cards become more common, people will carry more of them in their wallets. Anti-collision protocols become more important, so that readers are able to select the correct contactless card. The ISO 14443 standard provides anti-collision measures [29]. When a POS terminal detects multiple payment cards, it will likely not perform a transaction because there is no way to know with which card the cardholder wants to pay. For an attacker, it does not matter which payment card gets selected, as long as it is a payment card. However, if two different contactless cards are presented (e.g., a library card and a payment card), then both a POS terminal and an attacker will want to be able to select the payment card.

However, the tested Android devices were not able to select any card when three or more contactless cards are placed close to each other. When two contactless cards are close to each other, the Android devices simply communicates with the one that is most close by (even if it is not a payment card). This becomes a problem for an attacker if victims have more contactless cards close to each other in one wallet.

## 11.5 Research Timeline

We started our research by analyzing the documentation on EMV Contactless. We found out the EMV Contactless standard is very similar to the Contact standard. In many occasions, the books on EMV Contactless [13–22] refer to the books on EMV Contact [9–12] for more details. We knew that the EMV Contact specifications allow for many different options and parameters, so in order to get a good idea about what is used in the Netherlands, we needed to acquire many contactless cards and a contactless terminal, and perform tests on them.

At first we had some problems making contactless payments, especially with the newly acquired contactless cards. As it later turned out, this was because of the envelope fraud countermeasures that were implemented (before contactless transaction are possible, first a contact transaction with PIN had to be performed).

Because the EMV Contactless specifications are not very different from the EMV Contact specifications, we made the decision to not do a formal analysis of the EMV Contactless protocol, as the results could be predicted in advance. Instead, we focused on a possible relay attack, and ordered the necessary Android device. Google provided some helpful code samples to work with the NFC adapters in Android devices, so it only took us two days to realize a working relay setup.

Over time, we optimized our relay setup and introduced more features (such as a time delay to measure the limits of the POS terminal and the preloading functions).

## 11.6 EMV in the media

EMV has proven to be a lively topic. In April 2014, contactless payments were enabled and announced in the media[54]. Since the beginning of our research, in May 2014, EMV has appeared in multiple media reports. In May 2014, many Dutch news sites reported that researchers succeeded in 'cracking' the EMV chip[55]. However, the reported vulnerability was already known in 2012 [3].

In August 2014, RTL Nieuws made an item on critics claiming that the contactless cards are ideal for digital pickpockets. An attack is described where criminals would go around with a mobile POS terminal, hold it close to a contactless card of the victim and transfer €25 to their (or a money mule's) account. However, given that the account holder is easily traceable, and that the risk of being caught with a mobile POS terminal is significant, this attack does not look too practical.

[54] http://www.pin.nl/actueel/nieuws/ruim-4-miljoen-ing-betaalpassen-al-geschikt-voor-contactloos-betalen/

[55] https://www.security.nl/posting/388407/

Furthermore, in October 2014, the media reported that the 3 millionth contactless payment was performed[56]. Dutch Minister of Security and Justice, Ivo Opstelten, appealed to the Dutch people to perform less cash payments, as this will decrease the number of robberies.

[56] https://www.ing.nl/nieuws/nieuws_en_persberichten/2014/10/In_het_nieuws_contactloos_betalen_wint_vlot_terrein.aspx

# Chapter 12

# Future Work and Conclusions

This chapter provides a summary of our research and our findings. Section 12.1 gives an overview of the possible future work to further extend our relay attack and to find additional vulnerabilities. Section 12.2 gives answer to the research questions and presents all our conclusions.

## 12.1 Future Work

This section discusses possible future work. Section 12.1.1 describes how our relay attack could be improved with more research on the range of Android's NFC antenna. Section 12.1.2 gives some ideas on how to even further increase the speed of our relay attack, in case meaningful timing restrictions appear in POS terminals or in the back end. Section 12.1.3 describes some possible future research to make our relay attack work when multiple contactless cards are placed close to each other in a wallet. Section 12.1.4 describes possible research on the new features that EMV Contactless introduces, but that we could not investigate because they are not available in the Netherlands. Finally, Section 12.1.5 describes some additional research possible on the Dutch cards and POS terminals based on fuzz testing.

### 12.1.1 Extending the Range of Android's NFC

One of the limitations of our relay attack is the limited distance of the NFC range of Android devices (see Section 11.4.1). It would be very interesting to research where exactly the practical limit lies for the range of standard NFC devices. We already noticed an increase from approximately 1 cm to approximately 5 cm for Android devices the last four years. In laboratories, distances up to 27 cm are realized [23] and some researchers described methods to even further increase this to 55 cm [32]. If it is indeed possible to extend the range to such limits, with low cost hardware, this would be a very important extension to our relay attack,

making it possible to perform this attack on a large scale.

### 12.1.2 Improving Relay Transaction Time

The transaction times achieved with our relay setups are fast enough, because there are no countermeasures present in the tested POS terminals. However, if time restricting countermeasures were to be implemented, attackers would be able to even further improve the relayed transaction times by implementing the following three improvements:

- In our relay setups, we forward the entire `GENERATE AC` command from the Relay Device to the Mole Device. This entire command is 98 bytes, however, only 8 bytes (the UN) are *really* unpredictable. The day, amount, country code, etc., can be easily predicted or determined in advance. We noticed some small overhead decrease for smaller messages, but we did not investigate this further.

- Our relay setups use the TCP network protocol over the wifi adapter. However, the UDP network protocol is generally faster, so using UTP could decrease the overhead. Furthermore, using different adapters (such as Bluetooth or infrared), could also decrease the overhead of the transaction.

- If faster Android devices were used, the relayed transaction time of a typical Dutch card could be significantly lowered. The minimum timing results of the Relay Device differ 69 ms with the typical timing results (see Table 7.5). Furthermore, the difference between a fast computer and our Mole Device for issuing a `GENERATE AC` is very significant at 216 ms (see Tables 7.6 and 7.1).

### 12.1.3 Anti-Collision Protocol

In real world scenarios, multiple contactless cards are often stored together in a wallet. To improve the relay attack, it would be useful for the Mole Device to be able to communicate with a payment card that is close to other contactless cards. The ISO 14443 standard does provide anti-collision measures (see Section 11.4.2), however, in our experience these do not work at all. A good begin would be to be able to communicate with either card when two contactless cards are placed together in a wallet. If this could be extended to more contactless cards, then our relay attack would perform better in real world scenarios where multiple contactless cards are stored together in wallets.

### 12.1.4 Research New Features

The EMV Contactless specifications introduce many new features. Most of these features, especially those introduced by MasterCard, are not present in Dutch cards. Therefore, we were unable to research the security of the new features. In particular, the storage features introduce new attack factors, as incorrect or unexpected user input can be stored on the card, and will probably be read by the POS terminals. We already concluded that the most commonly used POS terminal in the Netherlands is not very robust against unexpected input (see Section 9.2). Furthermore, the on-device PIN verification is not used in the Netherlands, but could also be a security risk, since the result is not authenticated by the terminal for Contactless Mag-Stripe Mode transactions.

We were unable to find unattended contactless POS terminals that do not support PIN. However, they do exist[57]. It would be interesting to know if contactless cards can still pay at these terminals when already €50 has been payed contactlessly. Unfortunately, we were unable to test this. Furthermore, when more contactless POS terminals become available, it is very interesting to test these terminals too because we were only able to thoroughly test one.

### 12.1.5 Fuzz Testing on POS and Card

It is our expectation that more vulnerabilities will be identified when the cards and POS terminals are provided with large amounts of unexpected or incorrect input data. This can be performed in a structural way and is often called 'fuzz testing'. There are at least two vulnerabilities that came to light when input was used that was slightly different than expected. In particular, the offline PIN verification functionality was found on some cards when the command was issued before the right application was selected, and the Denial-of-Service on the POS terminal was found when a longer message was sent than expected. These findings are both accidental, as there was not searched for such vulnerabilities.

## 12.2 Conclusions

This section presents the research questions of this work and how they have been addressed. Furthermore, it gives an overview of all our findings.

**First Research Question:** *To what extent do exploitable vulnerabilities exist in the protocol specifications of EMV Contactless?*

The security features in EMV Contactless are optional to a great extent. As a result, we conclude that the EMV Contactless specifications allow both

[57] `http://paqar.nl/producten/vendingmachines/`

relatively secure and completely insecure implementations. There are very few security requirements given in the specifications, and issuers must ensure their implementation is secure.

**Second Research Question:** *To what extent do practically exploitable vulnerabilities exist in the Dutch implementation of EMV Contactless?*

We conclude that the Dutch implementations ensure that it is not possible to clone cards, to alter transaction details or to bypass PIN verification. The most important reasons are the policies to only allow online transactions and to not allow legacy modes. However, the Dutch implementation does not prevent relay attacks, which forms a risk for contactless cards. Furthermore, multiple vulnerabilities have been identified in specific card and POS terminal implementations, indicating that both the implementations and the certification processes are suboptimal. However, large scale attacks are not feasible (yet) because of the limited range of NFC in Android devices and the maximum amount limit for transactions performed without PIN.

**Overview** We extended existing research on relay attacks [32] and improved existing proofs-of-concept [33,39]. Our proof-of-concept is the first that only uses standard Android devices (instead of a laptop [39] and a BlackBerry phone [33]). Furthermore, we use EMV Contactless cards instead of Google Wallet [39] and dummy transactions [33]. In addition, this thesis is the first documentation of typical Dutch EMV Contactless timings.

With very limited resources (two stanAndroid devices and two days of programming), we immediately succeeded in performing a relay attack on all currently available Dutch EMV cards. With the use of EMV and Android specific optimizations in our proof-of-concept, we realized a relay attack that performs transactions with any Maestro card faster than a legitimate trdard ansaction performed directly with the slowest Maestro card.

Furthermore, this thesis is the first report of a relayed transaction on a contactless card that is faster than a transaction performed directly with that same card. This result indicates that the most obvious relay attack countermeasures, such as response time restrictions, will not be effective.

During the development of the proof-of-concept relay attack, we identified several other vulnerabilities. In particular, the Atos Worldline POS terminals crashed when we transmitted a longer message than they expected. This presumably is the result of a buffer overflow, which poses a serious security risk. One of the four banks issues cards that have no protection against envelope fraud, a term we introduced to describe the scenario where an attacker performs a relay attack on a contactless card that was sent via the mail. Furthermore, the Vodafone cards have a misconfiguration concerning their secret keys and transaction

counters. Due to the misconfiguration in the transaction counters, we found out that the secret symmetric keys are not unique and these cards thus break the security requirement on the uniqueness of the secret keys. We exploited the reuse of the secret symmetric keys to determine that the currency in cryptograms is not authenticated. Therefore, these cards do not use the recommended set of transaction data as defined by the EMV Contactless specifications. We did not search for these vulnerabilities, but we found them accidentally. These accidental findings and the general lack of countermeasures against relay attacks indicate that the certification processes of cards and POS terminals do not suffice. The absence of countermeasures against envelope fraud in only Knab cards indicates that the issuers do not share their security knowledge optimally.

We think that not using the PIN for small transactions is a good security practice, as this dramatically decreases the number of opportunities for shoulder surf attacks. With knowledge of the PIN, attackers can perform transactions up to €5,000. Furthermore, we propose to allow small PIN-less contact transactions, as a lost or stolen card already means that €50 can be stolen (contactlessly) anyway. However, as this might reduce the security perception of the users, banks could consider the option for users to decide the PIN-less limit (up to €50).

Furthermore, the limit for contactless transactions with PIN should be lowered. We do not think it is necessary to be able to pay €2,500 or €5,000 (depending on the bank) contactlessly. When a customer wants to pay such amounts, he probably would not mind using the chip card. However, using different maximum amounts for contact and contactless transactions could be interpreted as an implicit acknowledgment of the banks that contactless transactions are indeed less secure. Customers need to be confident that contactless transactions are very secure, otherwise they will not accept the new technology. Therefore, banks might not be willing to change the maximum contactless PIN limit.

The €25 limit per contactless PIN-less transactions does not add much security, as attackers can still steal €50 without knowing the PIN in two transactions. The €50 limit for consecutive contactless PIN-less transactions, however, is very effective and prevents that relay attacks can be performed with large profits.

# Bibliography


[1] Adida, Ben and Bond, Mike and Clulow, Jolyon and Lin, Amerson and Murdoch, Steven and Anderson, Ross and Rivest, Ron. Phish and chips. In *Security Protocols*, pages 40–48. Springer, 2009.

[2] Anderson, Ross and Bond, Mike and Murdoch, Steven J. Chip and spin. *Computer Security Journal*, 22(2):1–6, 2006.

[3] Mike Bond, Omar Choudary, Steven J Murdoch, Sergei Skorobogatov, and Ross Anderson. Chip and skim: cloning emv cards with the pre-play attack. *arXiv preprint arXiv:1209.2531*, 2012.

[4] Joseph Bonneau, Sören Preibusch, and Ross Anderson. A birthday present every eleven wallets? the security of customer-chosen banking pins. In *Financial Cryptography and Data Security*, pages 25–40. Springer, 2012.

[5] De Ruiter, Joeri and Poll, Erik. Formal analysis of the EMV protocol suite. In *Theory of Security and Applications*, pages 113–129. Springer, 2012.

[6] Dolev, Danny and Yao, Andrew C. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[7] Martin Emms, B Arief, T Defty, J Hannon, F Hao, and A Van Moorsel. *The dangers of verify PIN on contactless cards*. Computing Science, Newcastle University, 2012.

[8] Emms, Martin and Arief, Budy and Freitas, Leo and Hannon, Jospeh and van Moorsel, Aad. Harvesting high value foreign currency transactions from EMV contactless cards without the PIN. *School of Computing Science Technical Report Series*, (1421):5–12, 2014.

[9] EMVCo. EMV Contact Specifications for Payment Systems. In *Book 1: Application Independent ICC to Terminal Interface Requirements v4.3*. EMVCo, 2011.

[10] EMVCo. EMV Contact Specifications for Payment Systems. In *Book 2: Security and Key Management v4.3*. EMVCo, 2011.

[11] EMVCo. EMV Contact Specifications for Payment Systems. In *Book 3: Application Specification v4.3*. EMVCo, 2011.

[12] EMVCo. EMV Contact Specifications for Payment Systems. In *Book 4: Cardholder, Attendant, and Acquirer Interface Requirements v4.3*. EMVCo, 2011.

[13] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book A: Architecture and General Requirements v2.4*. EMVCo, 2014.

[14] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book B: Entry Point Specifications v2.4*. EMVCo, 2014.

[15] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book D: Contactless Communication Protocol v2.4*. EMVCo, 2014.

[16] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-1 Kernel 1 Specification v2.4*. EMVCo, 2014.

[17] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-2 Kernel 2 Specification v2.4*. EMVCo, 2014.

[18] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-3 Kernel 3 Specification v2.4*. EMVCo, 2014.

[19] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-4 Kernel 4 Specification v2.4*. EMVCo, 2014.

[20] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-5 Kernel 5 Specification v2.4*. EMVCo, 2014.

[21] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-6 Kernel 6 Specification v2.4*. EMVCo, 2014.

[22] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-7 Kernel 7 Specification v2.4*. EMVCo, 2014.

[23] Gerhard P Hancke. Practical attacks on proximity identification systems. In *Security and Privacy, 2006 IEEE Symposium on*, pages 6–pp. IEEE, 2006.

[24] ISO. Financial transaction card originated messages – Interchange message specifications – Part 1: Messages, data elements and code values. ISO 8583-1:2003, International Organization for Standardization, Geneva, Switzerland, 2003.

[25] ISO. Identification cards – Physical characteristics. ISO 7810:2003, International Organization for Standardization, Geneva, Switzerland, 2003.

[26] ISO. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 1: Physical characteristics. ISO 14443-1:2008, International Organization for Standardization, Geneva, Switzerland, 2008.

[27] ISO. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 2: Radio frequency power and signal interface. ISO 14443-2, International Organization for Standardization, Geneva, Switzerland, 2008.

[28] ISO. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 4: Transmission protocol. ISO 14443-4:2008, International Organization for Standardization, Geneva, Switzerland, 2008.

[29] ISO. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 3: Initialization and anticollision. ISO 14443-3:2011, International Organization for Standardization, Geneva, Switzerland, 2011.

[30] ISO. Identification cards – Integrated circuit cards – Part 1: Cards with contacts – Physical characteristics. ISO 7816-1:2011, International Organization for Standardization, Geneva, Switzerland, 2011.

[31] ISO. Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. ISO 7816-4:2013, International Organization for Standardization, Geneva, Switzerland, 2013.

[32] Ziv Kfir and Avishai Wool. Picking virtual pockets using relay attacks on contactless smartcard. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 47–58. IEEE, 2005.

[33] Konstantinos Markantonakis, Lishoy Francis, Gerhard Hancke, and Keith Mayes. Practical relay attack on contactless transactions by using nfc mobile phones. *Radio Frequency Identification System Security: RFIDsec*, 12:21, 2012.

[34] MasterCard. PayPass M/Chip Requirements, July 2013. [Available online at `https://www.paypass.com/PP_Imp_Guides/PayPass-MChip-Requirements-2013.pdf`; accessed 24th June 2014].

[35] MasterCard. Mastercard Best Practices For Mobile Point of Sale Acceptance, November 2014. [Available online at `http://www.mastercard.com/us/company/en/docs/MasterCard_Mobile_Point_Of_Sale_Best_Practices.pdf`; accessed 26th June 2014].

[36] McHugh, Sheli and Yarmey, Kristen. Near field communication. 2014.

[37] Medaglia, Carlo Maria and Moroni, Alice and Volpi, Valentina and Ceipidor, Ugo Biader and Vilmos, András and Benyó, Balázs. Services, Use Cases and Future Challenges for Near Field Communication: the StoLPaN Project. *Deploying RFID-Challenges, Solutions and Open Issues*, pages 265–290.

[38] Murdoch, Steven J and Drimer, Saar and Anderson, Ross and Bond, Mike. Chip and PIN is Broken. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 433–446. IEEE, 2010.

[39] Michael Roland. Applying recent secure element relay attack scenarios to the real world: Google wallet relay attack. *arXiv preprint arXiv:1209.0875*, 2012.

[40] Roland, Michael and Langer, Josef. Cloning Credit Cards: A combined pre-play and downgrade attack on EMV Contactless. *7th USENIX Workshop on Offensive Technologies*, pages 1–12, 2013.

[41] The UK Cards Association Limited. Standard 70 - Card Acceptor to Acquirer Interface Standards, 2013.

# Appendices

# Appendix A

# Maestro Type A EMV Mode Trace

Bytes that indicate the length of the content of a tag are underlined and not shown in the explanation of the command. Some bytes are anonymized with 'p' to hide privacy sensitive information. Some bytes are replaced with '....' where there are many bytes that do not add information (e.g. encrypted data or public keys). Commands from the terminal to the card are (briefly) explained in natural language, as it is more interesting what is commanded rather than how it is commanded. Responses from the card, however, are extensively described.

Terminal → Card (message 2 in Figure 3.4):
Select 2PAY.SYS.DDF01
`00A4 04 00 0E 325041592E5359532E4444463031 00`

Card → Terminal (message 3 in Figure 3.4):
9000 6F 2E 84 0E 325041592E5359532E4444463031 A5 1C 5F2D 04 6E6C656E
BF0C 12 61 10 4F 07 A0000000043060 87 01 01 9F2A 01 02

9000: Status OK
6F: File Control Information
  84: Dedicated File Name
    325041592E5359532E4444463031: 2PAY.SYS.DDF01
  A5: File Control Information Proprietary Template
    5F2D: Language Preference
      6E6C656E: nlen (Dutch, English)
    BF0C: File Control Information Issuer Discretionary Data
      61: Application Template
        4F Application Identifier
          A0000000043060
        87: Application Priority Indicator
          01
        9F2A: Kernel Identifier
          02

Terminal → Card (message 4 in Figure 3.4):
Select application with AID: A0000000043060
00A4 04 00 07 A0000000043060 00

Card → Terminal (message 5 in Figure 3.4):
9000 6F 1E 84 07 A0000000043060 A5 13 50 07 4D41455354524F 87 01 01 5F2D 04 6E6C656E

9000: Status OK
6F: File Control Information
  84: Dedicated File Name
    A0000000043060
  A5: File Control Information Proprietary Template
    50: Application Label
      4D41455354524F: MAESTRO
    87 Application Priority Indicator
      01
    5F2D Language Preference
      6E6C656E: nlen (Dutch, English)

Terminal → Card (message 6 in Figure 3.4):
Get Processing Options
80A8 00 00 02 83 00 00
Card → Terminal (message 7 in Figure 3.4):
9000 77 16 82 02 1980 94 10 08010100100101011801020020010200

9000: Status OK
77: Response Message Template Format 2
  82: Application Interchange Profile
    1980
  94: Application File Locator
    08010100: SFI 1, Record 1, not for offline authentication
    10010101: SFI 2, Record 1, use for offline authentication
    18010200: SFI 3, Records 1 and 2, not for offline authentication
    20010200: SFI 4, Records 1 and 2, not for offline authentication

Terminal → Card (message 8 in Figure 3.4):
Read Record SFI 1, Record 1
00B2 01 0C 00

Card → Terminal (message 9 in Figure 3.4):
9000 70 05 9F6C 02 FFFF

9000: Status OK
70: EMV Proprietary Template
  9F6C: Card Transaction Qualifiers
    FFFF

Terminal → Card (message 8 in Figure 3.4):
Read Record SFI 2, Record 1
00B2 01 14 00

Card → Terminal (message 9 in Figure 3.4):
9000 70 819C 5A 08 6706560000317518 5F24 03 190731 5F25 03 140701 5F28 02 0528 5F34 01 01 8C 21
9F02069F03069F1A0295055F2A029A039C019F37049F35019F45029F4C089F3403 8D 0C 910A8A0295059F37049F4C08 8E 0C 00000000000000000042031F03 9F07 02 FF00 9F08 02 0002 9F0D 05 B450848000 9F0E 05 0000180000 9F0F 05 B470848000 9F42 02 0978 9F4A 01 82
57 11 6706560000317518D19072011366002011

9000: Status OK
70: EMV Proprietary Template
  5A: Application Primary Account Number (PAN)
   6ppppppppppppppp8: PAN anonymized with 'p'
  5F24: Application Expiration Date
    190731: 31st July 2019
  5F25: Application Effective Date
    140701: 1st July 2014
  5F28: Issuer Country Code
    0528: The Netherlands
  5F34 : Application Primary Account Number (PAN) Sequence Number
    01
  8C: Card Risk Management Data Object List 1
    9F02069F03069F1A0295055F2A029A039C019F37049F35019F45029F4C089F3403
  8D: Card Risk Management Data Object List 2
    910A8A0295059F37049F4C08
  8E: Cardholder Verification Method List
    000000000000000042031F03
  9F07: Application Usage Control
    FF00
  9F08: Application Version Number
    02
  9F0D: Issuer Action Code  Default
     B450848000
  9F0E: Issuer Action Code  Denial
    0000180000
  9F0F: Issuer Action Code  Online
    B470848000
  9F42 : Application Currency Code
    0978: Euro
  9F4A: Static Data Authentication Tag List
   82
  57: Track 2 Equivalent Data
    6ppppppppppppppp8D190720113660020118: Track 2 Data (PAN anonymized with 'p')

Terminal → Card (message 8 in Figure 3.4):
Read Record SFI 3, Record 1
`00B2 01 1C 00`
Card → Terminal (message 9 in Figure 3.4):
`9000 70 81E0 8F 01 05 9F32 01 03 92 24 1F[34 additional bytes]33 90 81B0 3B[174 additional bytes]F4`

`9000`: Status OK
`70`: EMV Proprietary Template
  `8F`: Certification Authority Public Key Index
    `05`
  `9F32`: Issuer Public Key Exponent
    `03`
  `92`: Issuer Public Key Remainder
    `1F....33`: 36 bytes total
  `90`: Issuer Public Key Certificate
    `3B....F4`: 176 bytes total

Terminal → Card (message 8 in Figure 3.4):
Read Record SFI 3, Record 2
`00B2 02 1C 00`

Card → Terminal (message 9 in Figure 3.4):
`9000 70 03 93 01 FF`

`9000`: Status OK
`70`: EMV Proprietary Template
  `93`: Signed Static Application Data
    `FF`

Terminal → Card (message 8 in Figure 3.4):
Read Record SFI 4, Record 1
`00B2 01 24 00`

Card → Terminal (message 9 in Figure 3.4):
`9000 70 04 9F47 01 03`

`9000`: Status OK
`70`: EMV Proprietary Template
  `9F47`: Integrated Circuit Card (ICC) Public Key Exponent
    `03`

Terminal → Card (message 8 in Figure 3.4):
Read Record SFI 4, Record 2
`00B2 02 24 00`

Card → Terminal (message 9 in Figure 3.4):
`9000 70 81B4 9F46 81B0 27[174 additional bytes]EA`

`9000`: Status OK
`70`: EMV Proprietary Template
  `9F46`: Integrated Circuit Card (ICC) Public Key Certificate
    `27....EA`: 176 bytes total

Terminal → Card (message 2 in Figure 3.6):
Generate Application Cryptogram of type ARQC, with CDA performed
`80AE 90 00 2B 000000000001 000000000000 0528 0000008000 0978 140730 00 nnnnnnnn 21 0000 0000000000000000 3F0000 00`

`80AE`: `GENERATE AC`
  `90`: AC type: ARQC with CDA
    `000000000001`: Amount, Authorized: 0.01
    `000000000000`: Amount, Other: 0.00
    `0528`: Terminal Country Code: The Netherlands
    `0000008000`: Terminal Verification Results

0978: Transaction Currency Code: Euro
140730: Transaction Date: 30th July 2014
00: Transaction Type: Default value
nnnnnnnn: Unpredictable Number
21: Terminal Type: Attended, merchant operated, online only
0000: Data Authentication Code
0000000000000000: ICC Dynamic Number
1F0302: Cardholder Verification Method Results

Card → Terminal (message 4 in Figure 3.6):
9000 77 81A2 9F27 01 80 9F36 02 000E 9F4B 8180 23[126 additional bytes]76
9F10 12 0110A040032200000000000000000000000FF

9000: Status OK
77: Response Message Template Format 2
  9F27: Cryptogram Information Data
    80: AC type: ARQC
  9F36: Application Transaction Counter
    000E: 14th transaction
  9F4B: Signed Dynamic Application Data
    23....76: 128 bytes total
  9F10: Issuer Application Data
    0110A040032200000000000000000000000FF

# Appendix B

# Maestro Type B EMV Mode Trace

Bytes that indicate the length of the content of a tag are underlined and not shown in the explanation of the command. Some bytes are anonymized with 'p' to hide privacy sensitive information. Some bytes are replaced with '....' where there are many bytes that do not add information (e.g. encrypted data or public keys). Commands from the terminal to the card are (briefly) explained in natural language, as it is more interesting what is commanded rather than how it is commanded. Responses from the card, however, are extensively described.

Terminal → Card (message 2 in Figure 3.4):
Select 2PAY.SYS.DDF01
`00A4 04 00 0E 325041592E5359532E4444463031 00`

Card → Terminal (message 3 in Figure 3.4):
9000 6F 2C 84 0E 325041592E5359532E4444463031 A5 1A BF0C 17 61 15 4F 07
A0000000043060 50 07 4D41455354524F 87 01 01

9000: Status OK
6F: File Control Information
  84: Dedicated File Name
    325041592E5359532E4444463031: 2PAY.SYS.DDF01
  A5: File Control Information Proprietary Template
    BF0C: File Control Information Issuer Discretionary Data
      61: Application Template
        4F Application Identifier
          A0000000043060
        50: Application Label
          4D41455354524F: MAESTRO
        87: Application Priority Indicator
          01

Terminal → Card (message 4 in Figure 3.4):
Select application with AID: A0000000043060
00A4 04 00 07 A0000000043060 00

Card → Terminal (message 5 in Figure 3.4):
9000 6F 1E 84 07 A0000000043060 A5 13 50 07 4D41455354524F 87 01 01 5F2D 04 6E6C656E

9000: Status OK
6F: File Control Information
  84: Dedicated File Name
    A0000000043060
  A5: File Control Information Proprietary Template
    50: Application Label
      4D41455354524F: MAESTRO
    87 Application Priority Indicator
      01
    5F2D Language Preference
      6E6C656E: nlen (Dutch, English)

Terminal → Card (message 6 in Figure 3.4):
Get Processing Options
80A8 00 00 02 83 00 00
Card → Terminal (message 7 in Figure 3.4):
9000 77 0E 82 02 1980 94 08 0802030018010201

9000: Status OK
77: Response Message Template Format 2
  82: Application Interchange Profile
    1980
  94: Application File Locator
    08020300: SFI 1, Records 2 and 3, not for offline authentication
    18010201: SFI 3, Records 1 and 2, use Record 1 for offline authentication

Terminal → Card (message 8 in Figure 3.4):
Read Record SFI 1, Record 2
00B2 02 0C 00

Card → Terminal (message 9 in Figure 3.4):
9000 70 1F 9F42 02 0978 9F08 02 0002 57 13 6734000531570525297D19022010000010664F

9000: Status OK
70: EMV Proprietary Template
  9F42: Application Currency Code
    0978: Euro
  9F08: Application Version Number
    0002
   57: Track 2 Equivalent Data
        67340005ppppppp5297D19022010000010664F: Account number anonymized

Terminal → Card (message 8 in Figure 3.4):
Read Record SFI 1, Record 3
00B2 03 0C 00
Card → Terminal (message 9 in Figure 3.4):
9000 70 81E0 8F 01 05 9F32 01 03 92 24 5F[34 additional bytes]9F 90
81B0 A0[174 additional bytes]0B

9000: Status OK
70: EMV Proprietary Template
  8F: Certification Authority Public Key Index
    05
  9F32: Issuer Public Key Exponent
    03
  92: Issuer Public Key Remainder
    5F....9F: 36 bytes total
  90: Issuer Public Key Certificate
    A0....0B: 176 bytes total

Terminal → Card (message 8 in Figure 3.4):
Read Record SFI 3, Record 1
00B2 01 1C 00

Card → Terminal (message 9 in Figure 3.4):
9000 70 8181 5F25 03 140514 5F24 03 190228 9F07 02 3D00 5A 0A
67340005ppppppp5297F 5F34 01 01 8E 0C 00000000000000042031F03 9F0D 05
B450848000 9F0E 05 0000180000 9F0F 05 B470848000 8C 21
9F02069F03069F1A0295055F2A029A039C019F37049F35019F45029F4C089F3403 8D
0C 910A8A0295059F37049F4C08 5F28 02 0528 9F4A 01 82

9000: Status OK
70: EMV Proprietary Template
  5F25: Application Effective Date
    140514: 14th May 2014
  5F24: Application Expiration Date
    190228: 28th February 2019
  9F07: Application Usage Control
    3D00
  5A: Application Primary Account Number (PAN)
   67340005ppppppp5297F: PAN anonymized with 'p'
  5F34: Application Primary Account Number Sequence Number
    01
  8E: Cardholder Verification Method List
    00000000000000042031F03
  9F0D: Issuer Action Code  Default
     B450848000
  9F0E: Issuer Action Code  Denial
    0000180000
  9F0F: Issuer Action Code  Online
    B470848000
  8C: Card Risk Management Data Object List 1
    9F02069F03069F1A0295055F2A029A039C019F37049F35019F45029F4C089F3403
  8D: Card Risk Management Data Object List 2
    910A8A0295059F37049F4C08
  5F28: Issuer Country Code
    0528: The Netherlands
  9F4A: Static Data Authentication Tag List
    82

Terminal → Card (message 8 in Figure 3.4):
Read Record SFI 3, Record 2
`00B2 02 1C 00`

Card → Terminal (message 9 in Figure 3.4):
`9000 70 81BE 9F49 03 9F3704 9F47 01 03 9F46 81B0 2C[174 additional bytes]11`

`9000`: Status OK
`70`: EMV Proprietary Template
  `9F49`: Dynamic Data Authentication Data Object List
    `9F3704`: Unpredictable Number with length 4
  `9F47`: Integrated Circuit Card Public Key Exponent
    `03`
  `9F46`: Integrated Circuit Card Public Key Certificate
    `2C....11`: 176 bytes total

Terminal → Card (message 2 in Figure 3.6):
Generate Application Cryptogram of type Transaction Certificate, with CDA performed
`80AE 50 00 2B 000000000001 000000000000 0528 8000000080 0978 140626 00 nnnnnnnn 22 0000 0000000000000000 3F0000 00`

`80AE`: `GENERATE AC`
  `90`: AC type: ARQC with CDA
    `000000000001`: Amount, Authorized: 0.01
    `000000000000`: Amount, Other: 0.00
    `0528`: Terminal Country Code: The Netherlands
    `0000000080`: Terminal Verification Results
    `0978`: Transaction Currency Code: Euro
    `140626`: Transaction Date: 26th June 2014
    `00`: Transaction Type: Default value
    `nnnnnnnn`: Unpredictable Number
    `21`: Terminal Type: Attended, merchant operated, online only
    `0000`: Data Authentication Code
    `0000000000000000`: ICC Dynamic Number
    `1F0302`: Cardholder Verification Method Results

Card → Terminal (message 4 in Figure 3.6):
9000 77 81A2 9F27 01 80 9F36 02 0070 9F4B 81B0 81[126 additional bytes]74
9F10 12 2B11A040032230000000000000000000000FF

9000: Status OK
77: Response Message Template Format 2
  9F27: Cryptogram Information Data
    80: AC type: ARQC
  9F36: Application Transaction Counter
    00C5: 197th transaction
  9F4B: Signed Dynamic Application Data
    1B....6E: 128 bytes total
  9F10: Issuer Application Data
    2B11A040032200000000000000000000FF

# Appendix C

# Vodafone Card App 1 Trace

Bytes that indicate the length of the content of a tag are underlined and not shown in the explanation of the command. Some bytes are anonymized with 'p' to hide privacy sensitive information. Commands from the terminal to the card are (briefly) explained in natural language, as it is more interesting what is commanded rather than how it is commanded. Responses from the card, however, are extensively described.

Terminal → Card (message 1 in Figure 3.7):
Select 2PAY.SYS.DDF01
`00A4 04 00 0E 325041592E5359532E4444463031 00`

Card → Terminal (message 2 in Figure 3.4):
9000 6F 3F 84 0E 325041592E5359532E4444463031 A5 2D BF0C 2A 61 13 4F 07 A0000000032020 50 05 5620504159 87 01 01 61 13 4F 07 A0000000032010 50 05 5620504159 87 01 02

9000: Status OK
6F: File Control Information
  84: Dedicated File Name
    325041592E5359532E4444463031: 2PAY.SYS.DDF01
  A5: File Control Information Proprietary Template
    BF0C: File Control Information Issuer Discretionary Data
      61: Application Template
        4F Application Identifier
          A0000000032020
        50: Application Label
          5620504159: V PAY
        87: Application Priority Indicator
          01
      61: Application Template
        4F Application Identifier
          A0000000032010
        50: Application Label
          5620504159: V PAY
        87: Application Priority Indicator
          02

Terminal → Card (message 3 in Figure 3.7):
Select application with AID: A0000000032020
00A4 04 00 07 A0000000032020 00

Card → Terminal (message 4 in Figure 3.4):
9000 6F 3B 84 07 A0000000032020 A5 30 50 05 5620504159 87 01 01 9F11 01 01 9F12 05 5620504159 9F38 0C 9F66 04 9F02 06 9F37 04 5F2A 02 BF0C 08 9F5A 05 3109780528

9000: Status OK
6F: File Control Information
  84: Dedicated File Name
    A0000000032020
  A5: File Control Information Proprietary Template
    50: Application Label
      5620504159: V PAY
    87 Application Priority Indicator
      01
    9F11 Issuer Code Table Index
      01
    9F12 Issuer Code Table Index
      5620504159: V PAY
    9F38 Processing Options Data Object List
      9F66049F02069F37045F2A02
    BF0C File Control Information Issuer Discretionary Data
      9F5A: Application Program Identifier
        3109780528: 0978 = currency code for Euro,
                    0528 = country code for the Netherlands

Terminal → Card (message 5 in Figure 3.7):
Get Processing Options
80A8 00 00 12 83 10 B4A04000 000000000001 nnnnnnnn 0978 00

80A8: GET PROCESSING OPTIONS
  83: Command template
    B4A04000: Terminal Transaction Qualifiers
    000000000001: Amount, Authorized: 0.01
    nnnnnnnn: Unpredictable Number
    0978: Transaction Currency Code: Euro

Card → Terminal (message 6 in Figure 3.7):

9000 77 40 82 02 0000 9F36 02 0037 9F26 08 8F0285E6886F32CF 9F10 07 06111103A00000 57 10 4ppppppppppppppp4D170522012699237 5F34 01 00 9F6C 02 0000 9F6E 04 20000000

9000: Status OK
77: Response Message Template Format 2
  82: Application Interchange Profile
    0000
  9F36: Application Transaction Counter
    0037
  9F26: Application Cryptogram
    8F0285E6886F32CF
  9F10: Issuer Application Data
    06111103A00000
  57: Track 2 Equivalent Data
    4ppppppppppppppp4D170522012699237
  5F34: Application Primary Account Number Sequence Number
    00
  9F6C: Card Transaction Qualifiers
    0000
  9F6E: Form Factor Indicator
    20000000

# Appendix D

# Vodafone Card App 2 Trace

Bytes that indicate the length of the content of a tag are underlined and not shown in the explanation of the command. Some bytes are anonymized with 'p' to hide privacy sensitive information. Commands from the terminal to the card are (briefly) explained in natural language, as it is more interesting what is commanded rather than how it is commanded. Responses from the card, however, are extensively described.

Terminal → Card (message 1 in Figure 3.7):
Select 2PAY.SYS.DDF01
`00A4 04 00 0E 325041592E5359532E4444463031 00`

Card → Terminal (message 2 in Figure 3.4):
9000 6F 3F 84 0E 325041592E5359532E4444463031 A5 2D BF0C 2A 61 13 4F 07 A0000000032020 50 05 5620504159 87 01 01 61 13 4F 07 A0000000032010 50 05 5620504159 87 01 02

9000: Status OK
6F: File Control Information
  84: Dedicated File Name
    325041592E5359532E4444463031: 2PAY.SYS.DDF01
  A5: File Control Information Proprietary Template
    BF0C: File Control Information Issuer Discretionary Data
      61: Application Template
        4F Application Identifier
          A0000000032020
        50: Application Label
          5620504159: V PAY
        87: Application Priority Indicator
          01
      61: Application Template
        4F Application Identifier
          A0000000032010
        50: Application Label
          5620504159: V PAY
        87: Application Priority Indicator
          02

Terminal → Card (message 3 in Figure 3.7):
Select application with AID: A0000000032010
00A4 04 00 07 A0000000032010 00

Card → Terminal (message 4 in Figure 3.4):
9000 6F 3B 84 07 A0000000032010 A5 30 50 05 5620504159 87 01 02 9F11 01 01 9F12 05 5620504159 9F38 0C 9F66 04 9F02 06 9F37 04 5F2A 02 BF0C 08 9F5A 05 3109780528

9000: Status OK
6F: File Control Information
  84: Dedicated File Name
    A0000000032010
  A5: File Control Information Proprietary Template
    50: Application Label
      5620504159: V PAY
    87 Application Priority Indicator
      02
    9F11 Issuer Code Table Index
      01
    9F12 Issuer Code Table Index
      5620504159: V PAY
    9F38 Processing Options Data Object List
      9F66049F02069F37045F2A02
    BF0C File Control Information Issuer Discretionary Data
      9F5A: Application Program Identifier
        3109780528: 0978 = currency code for Euro,
                    0528 = country code for the Netherlands

Terminal → Card (message 5 in Figure 3.7):
Get Processing Options
80A8 00 00 12 83 10 B4A04000 000000000001 nnnnnnnn 0978 00

80A8: GET PROCESSING OPTIONS
  83: Command template
    B4A04000: Terminal Transaction Qualifiers
    000000000001: Amount, Authorized: 0.01
    nnnnnnnn: Unpredictable Number
    0978: Transaction Currency Code: Euro

Card → Terminal (message 6 in Figure 3.7):

9000 77 40 82 02 0000 9F36 02 0037 9F26 08 8F0285E6886F32CF 9F10 07 06111103A00000 57 10 4ppppppppppppppp4D170522012699237 5F34 01 00 9F6C 02 0000 9F6E 04 20000000

9000: Status OK
77: Response Message Template Format 2
  82: Application Interchange Profile
    0000
  9F36: Application Transaction Counter
    0037
  9F26: Application Cryptogram
    8F0285E6886F32CF
  9F10: Issuer Application Data
    06111103A00000
  57: Track 2 Equivalent Data
    4ppppppppppppppp4D170522012699237
  5F34: Application Primary Account Number Sequence Number
    00
  9F6C: Card Transaction Qualifiers
    0000
  9F6E: Form Factor Indicator
    20000000