EMV in a nutshell

Jordi van den Breekel[1], Diego A. Ortiz-Yepes[2], Erik Poll[3], and Joeri de Ruiter[3]

[1]KPMG
[2]IBM Research Zurich
[3]Radboud University Nijmegen

June 29, 2016

Abstract

EMV is the leading international standard for payment smartcards, used by over a billion cards worldwide. EMV is not a single protocol, but a large family of complex protocols, with many variants and configurations: it can be used at ATMs and point-of-sale terminals, for internet banking, and more recently also for contactless payments, including so-called mobile payments with NFC phones.

This report tries to give an overview of common EMV variants, with concise descriptions of the essence of these protocols and an overview of security limitations and weaknesses that have been reported over the years.

We decided to write this report because it is so hard to keep an overview of all the EMV variants: it is hard to grasp the essence of the protocols from the – long and complex – official specifications. We also provide a guide to the scientific literature on EMV.

Contents

Glossary

- AAC - Application Authentication Cryptogram
- AC - Application Cryptogram, which can be an AAC, ARCQ, or TC; for readability we simply write 'cryptogram' in this document
- AFL - Application File Locator; identifies files on the card, and indicates whether their content is included in the SSAD
- AIP - Application Interchange Profile; indicates which authentication options the card supports
- ARQC - Authorisation Request Cryptogram
- ATC - Application Transaction Counter
- CAM - Card Authentication Method
- CDA - Combined Data Authentication
- CDOL - Card Risk Management DOL
- CID - Cryptogram Information Data; indicates the type of the cryptogram and the actions to be performed by the terminal
- CVC - Card Verification Code
- CVM - Cardholder Verification Method
- dCVV - dynamic Card Verification Value
- DDA - Dynamic Data Authentication
- DDOL - Dynamic Data Authentication DOL
- DOL - Data Object List
- DRDOL - Data Recovery DOL
- IAD - Issuer Application Data; proprietary data to be sent to the issuer
- MAC - Message Authentication Code
- PAN - Primary Account Number
- PCII - POS Cardholder Interaction Information
- PDOL - Processing Options DOL
- PPSE - Proximity Payment System Environment
- PSE - Payment System Environment

- SDA - Static Data Authentication
- SSAD - Signed Static Application Data; used in SDA
- SDAD - Signed Dynamic Application Data; used in DDA and CDA
- TC - Transaction Certificate
- TDHC - Transaction Data Hash Code; used in CDA
- TDOL - Transaction Certificate DOL

Chapter 1

Introduction

EMV is the leading international standard for payments using smartcards, also called chip cards or ICCs (Integrated Circuit Cards). The initiative for EMV was taken by Europay, MasterCard and Visa in the 1990s, with the view to replacing magnetic stripe cards (aka magstripe or swipe cards) with smartcards. The first EMV specifications were released in 1996. Note that two decades later, EMV smartcards still have a mag-stripe for backwards compatibility, and in the USA the adoption of EMV is still an ongoing process in 2016.

EMV has been a success in reducing skimming fraud, as can be seen from the fraud statistics for the UK[1] and those for the Netherlands, which are summarised in Figure 1.1. Of course, as long as EMV smartcards still contain a mag-stripe, for backward-compatibility, criminals can still use skimmed card data in countries that have not migrated to EMV. One effective measure against this is has been to block use of cards in such countries, called *geo-blocking*.

The EMV standard is maintained by EMVCo, a company jointly owned by MasterCard, Visa, American Express, and JCB. According to `www.thatsemv.com`, a website of one of the manufacturers of EMV cards, 45% of payment cards worldwide are EMV cards, which amounts to over 1.6 billion cards, and 76% of payment terminals are EMV-compliant.

year	fraud (in M€)
2007	15
2008	31
2009	36
2010	20
2011	39
2012	29
2013	6.8
2014	1.3
2015	1.7

Figure 1.1: Skimming fraud in the Netherlands, where EMV was introduced in 2011, and geo-blocking in 2012[2].

The EMV standards

EMV specifies data formats and protocols for the interaction between a smart and a terminal. A terminal can be an ATM, Point-of-Sale (POS) terminal, or, for a hand-held internet banking token. In official EMV jargon, a terminal is called a Card Acceptance Device (CAD), but in this document we will simply use the term terminal. The EMV specs built on top of other standards, namely ISO/IEC 7816 for contact smartcards and ISO/IEC 14433 for contactless smartcards (aka proximity cards).

It is wrong to think of EMV as a single protocol. Instead, it specifies buildings blocks which can be combined and configured in many ways to construct protocols. Indeed, EMV smartcards commonly contain *several* software applications (called *applets*) that support different EMV variants for different purposes. For instance, a bank card might contain one applet for cash withdrawals at ATMs, another for payments at a point-of-sale, and a third for internet banking.

[1] Published by the UK Cards Association (`https://www.theukcardsassociation.org.uk`).

[2] Source: the Dutch Banking Association (`https://nvb.nl`) and the Dutch Payment Association (`https://www.betaalvereniging.nl`).

Large parts of the EMV specs are public, but some are proprietary. And even if the core standards are public, the precise configurations typically are not. The specs for EMV contact cards are written in four documents available from the EMVCo website [21–24]. The specifications for contactless cards are given in a further ten books, also available from the EMVCo website [20, 26–34]. These specs for contactless cards are less uniform that for contact cards: for contact cards there is one common set of documents for all brands of EMV cards, for contactless cards there is a common set of three books [20, 26, 34], and then a so-called *kernel specifications* for the specific card scheme, such as American Express, Discover, JCB, MasterCard, UnionPay, or Visa.

The specs of EMV-CAP, the EMV variants for internet banking and online payments, are not public. EMV-CAP was an initiative by MasterCard to use smartcards for online two-factor authentication. Here the smartcard is used in combination with a hand-held smartcard reader that has a small display and a (numeric) keypad. CAP stands for Chip Authentication Program.

Understanding the EMV protocols from the official specifications, insofar that these are public, is far from trivial. One issue is the sheer size. The EMV contact specs are over 700 pages. The common core specs of EMV contactless over 400 pages, and the additional kernel specs add up to over 1200 pages. Apart from the length of the documents, there are many options and parameters, and it is hard to keep an overview of the ways these interact. Moreover, the specs typically do not discuss security goals or try to describe to the essence of the protocols at some higher level of abstraction that the raw bits and bytes.

More accessible descriptions of EMV protocols can be found in the scientific literature, e.g. [3, 17, 18, 46, 54]. We decided it was useful to collect such descriptions in one document. Starting point for this document, in addition to the papers mentioned above, have been descriptions of EMV protocols that we wrote up in the course of our own research [9, 12, 14, 47, 49, 53].

Scope of this report

The overview of EMV in this report is incomplete in that it only discusses the more common EMV variants that we and other researchers have encountered over the years. In particular, we only discuss two of the seven variants (aka kernels) for EMV contactless.

We do not discuss the specifics of EMV contactless payments using NFC mobile phones. As far as the interaction with the terminal is concerned, such payments are identical to one of the variants of EMV contactless. In fact, if a Secure Element in the phone is used, this chip in the phone plays the same role as the chip in a normal bank card. Only in case that HCE (Host Card Emulation) is used, which means that the main processor in the phone rather than a separate chip is performing the transaction, is there essential difference with card-based transactions: even then the transaction is the same, but the key management is different, as short-lived symmetric keys are used, which are only valid for a single transaction. For an overview of different ways to realise NFC payments with mobile phones, see [47].

Another aspect of EMV we do not discuss is EMV Tokenisation [25]. This is the process by which the sensitive account number (or PAN, for Primary Account Number) that is used in EMV transactions is replaced with a temporary, less valuable number. For a discussion of EMV Tokenisation we refer to [48].

High level reflection on the design

Before we dive into the details of the EMV protocols, we want to start with some high level reflection of the design.

The age of EMV shows. EMV transactions still ultimately rely on symmetric cryptography (and then typically 3DES rather than AES). Given the capabilities of modern smartcards, using asymmetric crypto would make more sense, and provide stronger security guarantees. Indeed, the more modern variants of EMV use asymmetric crypto, but then in addition to rather then instead of symmetric crypto.

More recent standards for chip cards, for example the international standards for electronic passports [41], and the (quite similar) standard for electronic driving licenses (ISO18013) look

far more modern. Such protocols use the notion of a secure channel, which eliminates a lot of potential for problems, as discussed. Also, e-passports include support for authentication of the terminal by the card, something which EMV protocols do not support.

The fact that EMV does not use some notion of a secure channel is a root cause of a lot of problems. It gives a lot of scope for Man-in-the-Middle attacks, as the attacker can reorder, replay, and change individual messages. With a secure channel this would not be possible. Moreover, such a channel would automatically authenticate all messages and rule out some of the known attacks on EMV, which abuse the fact that some responses of the card – notably reporting whether the PIN code is correct – are not authenticated. The EMV specs do include the specification of secure tunnel, so-called Secure Messaging, but this is not used in normal EMV transactions, but only for interaction between the card and the issuing bank outside normal transactions.

Not surprisingly, backward compatibility between the different protocol variants has proved a source of problems. This is the case in (1) the backward compatibility of contact cards with magstripe (discussed on page 1), (2) the forced fallback from encrypted to unencrypted two offline PIN mode ([6] (discussed on page 17), and (3) the backward compatibility of different types of contactless cards with magstripe cards ([40] and [51], both discussed in Section 4.2.2).

Chapter 2

EMV Contact

This chapter describes the original EMV protocols for ISO/IEC 7816 contact smartcards. This description is largely based on [14].

Section 2.2 explains the basics of the EMV set-up and some fundamentals concepts used. Then Section 2.3 describes the steps that make up a transaction with an EMV contact card. At the end of this chapter, Section 2.4 discusses known weaknesses and attacks on EMV. But we start with an overview of the academic literature on EMV contact cards.

2.1 Overview of literature on EMV Contact

A brief overview of EMV is given in [15], but this mainly discusses the cryptographic primitives and the key infrastructure, and does not go into details about the actual protocol. EMV security is analysed in [54], but only with the view to using it for internet payments.

Researchers at the University of Cambridge have been studying EMV for the past decade. An overview of this research is given in [4]. This research has for instance looked at the EMV protocols and the associated liability shift [3, 5]. The most famous paper to come out of this research, entitled 'Chip and PIN is broken' [46], demonstrated a weakness in some EMV configurations where a Man-in-the-Middle attack can fool a terminal in accepting a payment without PIN. More recently, they demonstrated an attack which exploited the use of a weak, predictable number generators in some ATMs [10]. Researchers at the University of Cambridge also looked at the security of the EMV API in HSMs (Hardware Security Modules) [2] and the tamper-resistance of Point-of-Sale terminals [16].

Our own research into EMV started with an attempt to make a formal model of EMV. For this we used F#: we demonstrated that EMV Contact could be formalised in 700 lines of F# code [13, 14] and that this formal model could be analysed with the protocol verification tool ProVerif [8] in combination with FS2PV [7]. This verification inevitably revealed the known security weaknesses of EMV, but did not report new ones. In an effort to systematically look for implementation bugs we then investigated the use of state machine inference to automatically reverse-engineer EMV implementations [1]. We showed it is easy to obtain protocol state machines from cards by black box testing, using standard tools for learning state machines. This revealed a surprising variety between EMV implementations on different cards, but no security flaws.

2.2 EMV Basics

The EMV specs for contact smartcards are given in four books [21–24]. These define a highly configurable tool-kit for payment protocols, offering a choice between:

- three *Card Authentication Methods*: SDA, DDA, and CDA;

- five *Cardholder Verification Methods*: none, signature, online PIN, offline unencrypted PIN, and offline encrypted PIN;
- two types of transactions: on- and off-line transactions.

Many of these options are again parameterised, possibly using proprietary data and formats. For example, in the case of an online transaction, where the issuer has to authorise the card to complete a transaction, the EMV specs do not prescribe how this works, and every issuer can implement their own proprietary protocol solution.

To determine which EMV applications are available on a card, the terminal can read out the *Payment System Environment* (PSE). This is a list of EMV applications with corresponding priority that can be found in the file 1PAY.SYS.DDF01 on a card.

Key Infrastructure

The essence of the key infrastructure used in EMV is as follows:

- Every card has a unique symmetric key MK_{AC} that it shares with the issuer. This card is derived from the issuer's master key MK_I using some key diversification scheme.

  Using this key a session key SK_{AC} can be computed, based on the Application Transaction Counter (ATC). The ATC is a simple counter on the card, which is increased on every interaction and included as a nonce in security-critical steps protocol steps to prevent replays.
- The issuer has a public-private key pair (P_I, S_I), and the terminal knows this public key P_I.
- Cards that support asymmetric crypto also have a public-private key pair (P_{IC}, S_{IC}) and an associated certificate, signed by the issuer (or the owner of the card scheme).

Most cards nowadays are capable of asymmetric crypto. That EMV still relies on the symmetric keys to a large extent is essentially a legacy issue: at the time the standard was first developed, supporting asymmetric crypto was still a considerable cost factor for smartcards.

This key set-up provides cards with two mechanisms to prove authenticity of data:

1. All EMV cards can provide MACs (Message Authentication Codes) on messages, using the symmetric keys they share with the issuer. The issuer can check these MACs to verify authenticity of the messages, but the terminal cannot.

   These MACs are included in the Application Cryptograms (ACs) that the card provides as evidence that it has take some steps in a transaction.
2. Cards that support asymmetric crypto can also digitally sign messages. Not only the issuer can then check the authenticity of these messages, but also the terminal.

2.2.1 Card Authentication Mechanism (CAM)

The EMV standard defines three card authentication mechanisms: SDA, DDA, and CDA.

- **SDA (Static Data Authentication)**. For SDA the card provides some digitally signed data (including e.g. the card number and expiry date) to the terminal to authenticate itself. This allows the terminal to check the authenticity of this data, since it knows the issuer's public key. However, this does not rule out cloning. SDA cards are just as susceptible to cloning as mag-stripes.
- **DDA (Dynamic Data Authentication)**. DDA cards can do asymmetric crypto and have a public/private key pair, and a signature of the public key to prove its authenticity.

  DDA then involves a challenge-response mechanism, where the card proves its authenticity by signing a challenge chosen by the terminal using a private asymmetric key. Unlike SDA,

this does rule out cloning. The price for this is that a more expensive cards is needed, namely one that can do asymmetric crypto.

A design flaw in DDA is that after authentication of the card, the subsequent transaction is not tied to that authentication, and only uses the card's symmetric key. In other words, the terminal can authenticate the card but cannot verify that the subsequent transaction is actually carried out by that card.

- **CDA (Combined Data Authentication)**. CDA repairs this deficiency of DDA. With CDA the card digitally signs all important transaction data, not only authenticating the card, but also authenticating the transaction.

2.2.2 DOL (Data Object List)

An important concept in the EMV specification is that of *Data Object List*, or *DOL*. A DOL specifies a list of data elements, and the format of such a list. Example data elements are the card's *Application Transaction Counter (ATC)*, the transaction amount, the currency, the country code, and card- or terminal-chosen nonces.

An EMV card supplies several DOLs to the terminal. Different DOLs specify which data the card expects as inputs in some protocol step (and then also the format that this data has to be in) or which data the card will provide as (signed) output in some protocol step. This is explained in more detail in Section 2.3.

The use of these DOLs make EMV highly parameterisable. The choices for DOLs are of crucial importance for the overall security, as they control which data gets signed or MACed, so no security analysis is possible without making some assumptions on the DOLs. For example, leaving out nonces of even the transaction amount from the DOL for the payment confirmation would obviously result in an insecure protocol.

2.3 An EMV Protocol Session

An EMV protocol session can roughly be divided into four steps:

1. *initialisation*: selection of the application on the smartcard and reading of some data;
2. (optionally) *data authentication*, by means of SDA, DDA, or CDA;
3. (optionally) *cardholder verification*, by means of PIN or signature;
4. the actual *transaction*.
5. (optionally) *issuer-to-card script processing*, which allows issuers to update cards in the field.

Each of these steps is described in more detail below. Here we use the usual semi-formal Alice-Bob style for security protocols, where square brackets indicate optional (parts of) messages. The card is denoted by C, and the terminal by T.

2.3.1 Initialisation

In the first phase of the EMV session, the terminal obtains basic information about the card (such as card number and expiry date) and the information about the features the card supports and their configurations. This is information the terminal needs for the subsequent steps in the EMV session.

Optionally, the card may request some information from the terminal before providing this information, as specified in the first response.

T → C: `SELECT_APPLICATION`
C → T: [PDOL]
T → C: `GET_PROCESSING_OPTIONS` [(data specified by the PDOL)]
C → T: (AIP, AFL)
Repeat for all records in the AFL:
T → C: `READ_RECORD` (i)
C → T: (Contents of record i)

The protocol starts by selecting the payment application. In response to the selection, the card optionally provides a *Processing Options Data Object List (PDOL)*. The PDOL specifies which data, if any, the card wants from the terminal; this could for instance include the Terminal Country Code or the amount.

The card then provides its *Application Interchange Profile (AIP)* and the *Application File Locator (AFL)*. The AIP consists of two bytes indicating the supported features (SDA, DDA, or CDA, offline PIN, and if so encrypted or plaintext, etc.) and whether terminal risk management should be performed.

The AFL is a list identifying the files to be used in the transaction. For each file it is indicated whether it is included in the offline data authentication; if so, the contents of this file is authenticated as part of the card authentication later.

The following files are mandatory for the card to have

- *Application Expiry Date*,
- *Application Primary Account Number (PAN)*,
- *Card Risk Management Data Object List 1 (CDOL1)*, and
- *Card Risk Management Data Object List 2 (CDOL2)*.

For cards that support SDA the *Signed Static Application Data (SSAD)* is also mandatory.

Note that none of the data provided by the card or by the terminal is authenticated at this stage. The process of card authentication, discussed below, will authenticate some of the data provided by the card.

2.3.2 Card Authentication

As already mentioned, there are three data authentication methods. The AIP tells the terminal which methods the cards supports, and the terminal should then choose the 'highest' method that both the card and itself support.

2.3.2.1 SDA

On cards that support SDA, the *Signed Static Application Data (SSAD)* is the signed hash of the concatenation of the files indicated by the AFL, optionally followed by the value of the AIP. Whether the AIP is included is indicated by the optional *Static Data Authentication Tag List*, which can be specified in a record. For SDA no additional communication is needed, since the data needed to verify the SSAD was already retrieved in the initialisation phase.

2.3.2.2 DDA

For DDA some additional communication is needed. DDA consists of two steps.

First, the certificate containing the card's public key is checked. This certificate also contains the hash of the concatenation of the files indicated by the AFL, so this authenticates the same static data that is authenticated with SDA.

Second, a challenge-response protocol is performed, where the card proves knowledge of the private key matching the public key in its certificate. For the challenge an `INTERNAL_AUTHENTICATE` message is sent to the card. The argument of this message is the data specified by the *Dynamic*

Data Authentication Data Object List (DDOL). The DDOL can be supplied by the card; otherwise a default DDOL should be present in the terminal. The DDOL always has to contain at least a terminal-generated nonce.

The *Signed Dynamic Application Data (SDAD)* is the signed *ICC Dynamic Data* and hash of the ICC Dynamic Data and data specified by the DDOL. The ICC Dynamic Data contains at least a time-variant parameter, e.g. a nonce or a transaction counter. On the bank and credit cards we studied, the DDOL only specified a terminal-generated nonce, and the ICC Dynamic Data only consisted of a card-generated nonce.

T → C: `INTERNAL_AUTHENTICATE`(data specified by DDOL)
C → T: $\mathsf{sign}_{S_{IC}}$(ICC Dynamic Data,
H(ICC Dynamic Data, data specified by DDOL))

We write H for hashing and sign for signing.

2.3.2.3 CDA

Unlike DDA, Card authentication using CDA does not require additional messages, but is part of the actual transaction. As with DDA, the Static Data to be Authenticated is authenticated using the certificate for the public key of the card.

With CDA, the Signed Dynamic Application Data (SDAD) provided during the transaction is a signature on a card-generated nonce, the CID, the cryptogram, the *Transaction Data Hash Code (TDHC)* and a terminal-generated nonce (specified by the CDOL1 and CDOL2):

$$\text{SDAD} = \mathsf{sign}_{S_{IC}}(\ \text{nonce}_{IC}, \text{CID, AC, TDHC,}\ \mathsf{H}(\text{nonce}_{IC}, \text{CID, AC, TDHC, nonce}_{Terminal})).$$

Here the TDHC is a hash of the elements specified by the PDOL, the elements specified by the CDOL1, the elements specified by the CDOL2 (in case of the second `GENERATE_AC` command) and the elements returned by the card in the response.

2.3.3 Cardholder Verification

Cardholder verification can be done in several ways: by a PIN, a handwritten signature, or it can simply not be done. The process to decide which *Cardholder Verification Method (CVM)* is used – if any – is quite involved. The card provides the terminal with its list of CVM Rules, that specify under which conditions which CVMs are acceptable, in order of decreasing preference. The terminal then chooses the CVM to be used. In all Dutch banking cards we inspected, the CVM List is included in the Static Data to be Authenticated.

If cardholder verification is done by means of a PIN, there are three options:

- online PIN, in which case the bank checks the PIN;
- offline plaintext PIN, in which case the chip checks the PIN, and the PIN is transmitted to the chip in the clear;
- offline encrypted PIN, in which case the chip checks the PIN, and the PIN is encrypted before it is sent to the card.

Note that the card is only involved in cardholder verification in case of offline PIN, either plaintext or encrypted, as detailed below. Encrypting the PIN requires a card that supports asymmetric crypto. Online PIN verification is performed using a different protocol between the terminal and the bank. In this case the card is not involved in the cardholder verification.

One possible reason to prefer online PIN over offline PIN is that it forces the terminal to go online and the issuing bank can then check if the card is reported stolen or missing. Note that this consideration does not have anything to do with the relative pros and cons w.r.t. protecting the confidentiality of the PIN code between checking it offline by the card vs checking it online against a central database.

2.3.3.1 Offline Plaintext PIN Verification

With plaintext PIN verification the PIN code is sent in plain to the card, which in its turn will return an *unauthenticated* response indicating whether the PIN was correct or how many failed PIN attempts there are left before the card blocks.

T → C: `VERIFY`(pin)
C → T: `Success`/ (`PIN_failed`, tries_left) / `Failed`

2.3.3.2 Offline Enciphered PIN Verification

If the verification is done using the encrypted PIN, the terminal first requests a nonce from the card. Using the public key of the card, the terminal encrypts the PIN together with the nonce and some random padding created by the terminal. The result of the verification is then returned unauthenticated to the terminal.

T → C: `GET_CHALLENGE`
C → T: nonce_{IC}
T → C: `VERIFY`($\mathsf{encrypt}_{P_{IC}}$(pin, nonce_{IC}, random_padding))
C → T: `Success`/ (`PIN_failed`, tries_left) / `Failed`

Here we write encrypt for encrypting.

2.3.4 The transaction

In the final step of an EMV session, after the optional card authentication and card holder verification, the actual transaction is performed. Transactions can be offline or online. The terminal chooses which it wants to use, but the card may refuse to do a transaction offline and force the terminal to do an online transaction instead.

For a transaction the card generates one or two cryptograms: one in the case of an offline transaction, and two in the case of an online transaction.

- In an offline transaction the card provides a proof to the terminal that a transaction took place by means of a *Transaction Certificate (TC)*, which the terminal sends to the issuer later.
- In an online transaction the card first provides an *Authorisation Request Cryptogram (ARQC)* which the terminal forwards to the issuer for approval. If the card receives approval, the card then provides a Transaction Certificate (TC) as proof that the transaction has been completed.

In both on- and offline transactions the card can also choose to refuse or abort the transaction, in which case an *Application Authentication Cryptogram (AAC)* is provided instead of TC or ARQC.

Below we first discuss the different types of cryptograms, before we describe the protocol steps for off- and online transactions.

2.3.4.1 Cryptograms

Using the `GENERATE_AC` command, the terminal can ask the card to compute one of the types of cryptograms mentioned above, i.e. TC, ARQC or AAC.

Arguments of the `GENERATE_AC` command tell the card which type of cryptograms to produce and whether CDA has to be used. Additional arguments that have to be supplied by the terminal are specified by CDOL1 and CDOL2. CDA is only performed on TC or ARQC messages, and not on AAC messages.

The response always contains

- the Cryptogram Information Data (CID),
- the Application Transaction Counter (ATC) and

- an Application Cryptogram (AC) or proprietary cryptogram.

Optionally, the response may contain

- the Issuer Application Data (IAD),
- other proprietary data,
- the Signed Dynamic Application Data (SDAD), namely if CDA is requested and the type of the response message is not AAC.

The cryptogram returned by the card can either be in the format specified in the EMV standard, or in a proprietary format. Since we do not know how the cryptogram is computed on the Dutch banking cards, we follow the recommendations from the EMV specification. Here a MAC is computed using the symmetric key SK_{AC}, which is shared between the card and the issuer, on a minimum set of recommended data elements. The recommended minimum set of elements to be included in the cryptogram is specified in Book 2, Section 8.1.1. It consists of the amount, terminal country, terminal verification results, currency, date, transaction type, terminal nonce, AIP and ATC. The AIP and ATC are data provided by the card, the other elements are provided by the terminal in the CDOL1 and CDOL2.

Additionally, if CDA is used, the card also returns the SDAD over the response using its private key P_{IC} so that the terminal can check the authenticity of the complete message.

Both CDOL1 and CDOL2 always include a terminal-generated nonce. A CDOL might request a *Transaction Certificate Hash*, which is a hash on the elements in the *Transaction Certificate Data Object List (TDOL)*. The TDOL might be provided by the card, or a default by the terminal can be used.

If no CDA is performed, the response to a `GENERATE_AC` command consists of the CID, the ATC, the cryptogram and optionally the IAD. When performing CDA, the cryptogram is replaced by the SDAD.

The `GENERATE_AC` command starts with a parameter indicating the type of AC that is requested. The boolean parameter cda_requested specifies whether a CDA signature is requested.

T $\rightarrow$ C: `GENERATE_AC` (TC, cda_requested, data specified by the CDOL1)
C $\rightarrow$ T: TC = (CID, ATC, MAC, [IAD])

where MAC = $\mathsf{MAC}_{MK_{AC}}$(amount, terminal country, terminal verification results, currency, date, transaction type, terminal nonce, AIP, ATC).

The card may refuse to do the transaction offline, and generate an ARQC cryptogram instead of the requested TC, forcing the terminal to go online.

In an online transaction the `EXTERNAL_AUTHENTICATE` command is used to pass the issuer's response to the ARQC back to the card. Alternatively, this response can be passed as a parameter of the next `GENERATE_AC` command.

2.4 Known attacks and weaknesses of EMV

EMV has several known security weaknesses and limitations:

1. **Mag-stripe cloning using chip information**

   A weakness in the design of EMV is that data obtained from the chip may be sufficient to reconstruct the mag-stripe. A good design would have made sure that some information needed for the mag-strip can not be obtained by the chip (in the same way that the CVC written on the back of a credit card is not included in the mag-stripe).

   Criminals have exploited this weakness in practice: in 2011 criminals were convicted in the Netherlands for placing manipulated EMV-CAP readers for internet banking (as discussed in Chapter 3) inside bank branches[1]. The bank provided facilities to customers to do internet banking in the bank branches. These customers could then be skimmed: the EMV-CAP readers use a plaintext offline PIN, so the PIN as well as the mag-stripe data could be eavesdropped using these devices.

2. **Reliance of symmetric keys for authenticity**

   An obvious and fundamental limitation of all versions prior to CDA is the reliance of symmetric crypto to authenticate transactions. The symmetric key for this has to be shared between the card and the back-end, so this does not provide true non-repudiation.

   Moreover, because the symmetric key is only shared between the card and the back-end, and the terminal does not have it, the terminal can not check the authenticity of cryptograms from the card. This is what causes some of the problems listed below.

3. **Cloning SDA cards**

   SDA cards can be cloned, and these clones can be used for offline transactions. When offline PIN verification is used, a clone can of course be programmed to approve any PIN, hence it is also called a yes-card. The issuer can spot this fraud, because the MACs will be incorrect, but the terminal cannot.

4. **Faking offline transactions with DDA cards**

   With a genuine DDA card, a terminal can still be fooled into accepting a fake offline transaction.

   Again, the root cause that makes this attack possible is that the terminal does not have the shared key used to compute the MAC in the cryptogram. It was clearly a flaw in the design of DDA: if the card can do asymmetric crypto and has a private key with an associated certificate, then there is no good reason why should not use this key to sign transactions.

   This attack is not so interesting from an attacker's point of view: the genuine card is still needed for the card authentication, and with access to the card one might as well then generate a genuine cryptogram. Also, the fake cryptogram can be detected as fake by the back-end.

5. **Faking offline PIN verification**

   A terminal can be fooled into thinking a PIN was correctly entered [46]. The root cause here is that the card's reply to say whether the PIN was correct is not authenticated. The cryptogram will reveal that the transaction was without PIN, so any online transaction could be blocked by the bank.

   This attack has been used in practice by criminals [36]. Here they did not only exploit that the response to the PIN verification cannot be authenticated by the terminal, but also that other responses from the card, in particular the ATC and the last ATC online, cannot be authenticated by the terminal. Presumably this was done to make sure that the risk management in the terminal decided for an offline transaction.

[1] See `http://deeplink.rechtspraak.nl/uitspraak?id=ECLI:NL:RBROT:2011:BU6142`

6. **Rollback to unencrypted PIN**

   A man-in-the-middle attack can force a rollback to unencrypted offline PIN, which then allows the plaintext PIN code to be observed [6]. In this attack the CVM list provided by the card is modified. Surprisingly, this attack may be possible even when the CVM list is signed, which it typically is; in this case the terminal can tell the CVM list has been modified, so it is strange that it will let the transaction proceed. It suggests that availability is a more important concern than security, assuming that this is a deliberate choice.

7. **Bad random-number generators**

   Bond et al. demonstrated an attack which exploited the use of a weak, predictable number generators found in some ATMs [10]. Here an attacker with temporary access to a card could harvest data by interacting with a card – collecting a large set of challenges and responses – that could be used at a later stage to fool an ATM with a bad random-number generator into accepting a transaction.

Chapter 3

EMV-CAP

EMV-CAP is a proprietary standard of Mastercard for using EMV banking cards for internet banking. It was also adopted by Visa, under the name Dynamic Passcode Authentication (DPA). This chapter summarises the publicly available information about EMV-CAP. Insofar as we know, this covers all the versions of EMV-CAP used in practice.

For EMV-CAP, an EMV smartcard is used in a hand-held reader that has a small display and a numeric keyboard. This then provides a second-factor to authenticate the user logging in to the bank's website or authorising transactions. The user enters his smartcard, provides his PIN code, and types in some challenge, after which the display provides a response which is derived from an EMV cryptogram. For EMV-CAP an additional application is included on the card next to the regular EMV application.

The EMV-CAP readers are typically stand-alone devices but there are some variants which are connected to a PC or laptop. This can reduce the amount of typing the user has to do; it allows for more meaningful text on the display on the device, which can help against phishing attacks and Man-in-the-Browser attacks. Some of these connected devices use a USB connection for bi-directional communication, for example the IBM ZTIC [55]. There are also devices that only use one-way communication from the laptop or PC to the reader using the display as an optical information channel. For instance, the German Volksbank uses the Sm@rtTAN-Optic, which uses a flickering barcode to transmit information. A newer solution is the CrontoSign[1] technology, which uses a coloured pattern of dots, a bit like a coloured version of a QR code, to transfer information. This is used by Vasco's Digipass Authenticator, used by the Dutch Rabobank (under the name Rabo Scanner).

3.1 Overview of the literature on EMV-CAP

EMV-CAP was first reverse-engineered by Drimer et al. [17], who studied versions of EMV-CAP used by British banks. Szikora and Teuwen then went on to reverse-engineer other variants, based on observations of EMV-CAP readers issued by Belgian banks [52]. They also provide a software emulation of EMV-CAP[2]. An important source of information they used is an online document [11] that describes different modes of EMV-CAP in an appendix, namely Mode 1, Mode 2, Mode 2 + TDS, and Mode 3. This seems to be the official terminology, as it appears in numerous documents and on websites of several companies. The research by Szikora and Teuwen revealed a security flaw in Mode 2 + TDS, where a *constant challenge* – consisting of all zeroes – is sent to the card, even though the user has typed in a challenge on the EMV-CAP reader; The challenge *is* used later in the subsequent computation of the response shown on the display, but by sending a fixed challenge to the card it is possible for an attacker to harvest responses from a card that could be used for internet banking later.

[1] `www.cronto.com`

[2] Available from `http://sites.uclouvain.be/EMV-CAP`

We confirmed the descriptions of these protocols for smartcard readers used by Dutch banks. We also investigated a USB-connected card reader for internet banking produced by Gemalto used by a Dutch bank, which uses a slightly different version of EMV-CAP. This analysis revealed an embarrassing security flaw where malware on the PC could force the card reader to complete security critical actions without the user's cooperation [9].

3.2 EMV-CAP Modes

For an EMV-CAP session, a full EMV transaction is performed by first requesting an ARQC followed by a request for an AAC. This is basically an online EMV transaction that is aborted. A session starts by selecting the EMV-CAP applet, reading some data from the card, including CDOL1 and CDOL2, and requesting the PIN from the user. The user is optionally presented with a challenge and/or other transaction-related data by the bank. After the PIN verification, this data is either entered on the hand-held reader by the user or transferred to it using, for example, a USB connection. The ARQC is used in combination with the ATC to generate a response for the bank, while the AAC is only used to leave the card in a clean state by correctly completing the transaction.

The response is typically computed by applying the Issuer Proprietary Bitmap (tag 0x9F56, called the CAP bit filter in [17]) on the concatenation of the PSN, ATC, ARQC and IAD, thus selecting only part of the bits from these values. The PSN is optionally included in the bitfilter and whether this is the case is indicated in the Issuer Authentication Flag (tag 0x9F55) on the card. The result of this bitfilter is then displayed to the user after being converted to digits, making it easier for the user to enter them again on the bank's website. Several modes exist within the standard, defining what data is sent to the card in the GENERATE AC commands and how the response is computed.

From the perspective of the card, all modes look the same: the card performs a regular EMV transaction that is aborted. The differences are in which parts of the data sent along with the GENERATE AC command are fixed or contain transaction-specific data, such as the Unpredictable Number (UN) or Authorized Amount, and how the response is computed by the reader.

3.2.1 Mode 1 and Mode 3

In Mode 1 the bank always provides a challenge to the user. This challenge is then sent to the card with the GENERATE AC command in the field for the Unpredictable Number, as specified in the corresponding CDOL. Next to this, the amount and currency code can also be included in the command in the Authorized Amount and Transaction Currency fields respectively. If a value is not included it is replaced with a constant value of all zeroes. The response is computed by applying the bitfilter as discussed before. A typical example of Mode 1 without amount or currency code can be seen in Figure 3.1. This variant, where no amount or currency code is set, is also referred to as Mode 3.

3.2.2 Mode 2

Mode 2 is identical to Mode 1, except that no challenge or additional data is included in the GENERATE AC commands at all, so no user input is required apart from the PIN. The response value is therefore only dependent on the ATC, which makes it possible to harvest these responses in advance from a card. This could only be detected by the bank, namely if an authorisation using a higher ATC already took place before the harvested responses are used.

3.2.3 Mode 2 + TDS

Mode 2 + TDS (Transaction Data Signing) extends Mode 2 with the possibility to sign additional data from the bank in the authorisation. The different data fields from the bank are concatenated

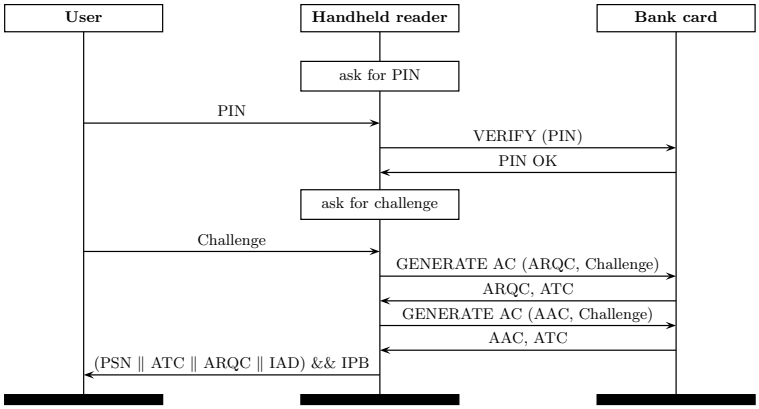

Figure 3.1: A typical EMV-CAP session using Mode 1 without amount or currency code

and used as input to a CBC-MAC using DES. The key that is used for this MAC is the ARQC, and the resulting MAC is used instead of the ARQC when applying the bitfilter to compute the response.

As with the regular Mode 2, no transaction related data is sent to the card. So again, data can be harvested in advance and then used at a later stage to compute the correct response when the bank presents the challenge and additional data to be signed. This is all the more surprising, because the user does enter input on the device. The protocol would be more secure, if instead of all zeroes, some dynamic data derived from the user input would be sent to the smartcard, thus tying the cryptogram to a particular transaction and making the harvesting of cryptogram impossible.

3.3 Known attacks and weaknesses of EMV-CAP

- **Cross channel vulnerabilities**

  Usually, the PIN code used for the EMV-CAP application on a bank card is identical to the one used for the regular EMV transaction. This is convenient for the user, as he only needs to remember one PIN, but introduces a security risk as well [17]. For example, when using a rogue point-of-sale terminal to pay for your groceries, this terminal could at the same time contact your bank and set up an online transfer. After all, with access to the card and knowledge of the PIN, the terminal can to complete EMV-CAP transactions and let the card compute the correct responses to authorise transactions.

  A typical measure banks take to prevent this kind of attack is by asking the user for additional data that is not present on the chip of the bank card when authorising an action, for example when logging in to the bank's website. This additional data could for example be a username/password combination or some number on the card that is not provided electronically via the smartcard (like the CVC code on the back of the credit cards).

- **Fixed challenge in Mode 2 + TDS**

  An obvious weakness in Mode 2 + TDS pointed out in [52] is that this protocol involves a fixed challenge, of all zeroes. The used of a fixed, predictable challenge means that responses could be harvested from a card and then used for internet banking at some later stage. Such a basic design flaw is all the more surprising because the user does type in a random challenge

on the EMV-CAP reader during this transaction: why is this random challenge not used as the challenge sent to the smartcard?

- **Overloaded semantics**

  Drimer et al. discuss the possible risk of overloading the semantics of particular EMV-CAP interactions [17]. For example, Mode 1 might be used both for logging in and signing transactions, where for logging in the transaction amount is set to 0. As a result of this, logging in and transferring zero euros result in an identical response code. An attacker might therefore use social engineering to convince a user to perform a transaction of zero euros and reveal the response code, thus enabling the attacker to log in on the user's account using this response code.

- **Lack of understandable semantics for the user**

  The main limitation of EMV-CAP is that the user does not really know what he is exactly signing. For instance, with Mode 1, when the user is provided with a challenge and amount of the transaction, the destination account of the transaction is not included in the authorisation. The user can therefore never be sure where he is sending the money to. This is even worse when the amount is not included in the authorisation, as he cannot even be sure how much money is being transferred.

  When using Mode 2 + TDS, additional data can be included in the authorisation. However, the meaning of these data fields is not standardised and it depends on the implementation of the hand-held reader how clear their meaning is to the user. The only way to indicate the meaning is therefore through the website, but that might be under control of the attacker, in case of a Man-in-the-Browser attack.

  In practice, criminals have used this weakness in social engineering attacks to by-pass the additional security offered by EMV-CAP readers. One attack here is to phone victims, pretending to be a bank employee, and then ask for EMV-CAP codes to perform some special security check. There is then no way the victims can sport that the codes they are providing are actually codes for authorising bank transfers. Phishing attacks by email or via malware can be used to find victims and obtain their telephone numbers.

3.4 Usage of EMV-CAP and some variants

Drimer et al. describe the EMV-CAP modes used by NatWest and Barclays in the UK [17]. Both banks use Mode 1, but Barclays also uses Mode 2 to log in. EMV-CAP readers issued by these banks are compatible: they have buttons for three operations: *identify*, *respond*, and *sign*.. Two fields used for the cryptogram generation are Authorized Amount (AA, tag 0x9F02) and Unpredictable Number (UN, tag 0x9F37):

1. *identify* corresponds with Mode 2, with AA and UN both fixed to zero.
2. *respond* corresponds with Mode 1, with AA zero and UN the challenge.
3. *sign* corresponds to Mode 1, with AA the amount and UN the destination account number.

All other fields have hard-coded default values, which are hard-coded in the reader and identical for NatWest and Barclays. For all options, the response of the device is obtained by applying the bit filter to the ARQC returned by the card, providing an 8 digit decimal response.

Variants of EMV-CAP used by Belgian banks are discussed in [52]. Here also Mode2 + TDS is used. This mode are also used by the Dutch Rabobank, their so-called Random Reader.

The e.dentifier2 of the Dutch ABN-AMRO bank, a device produced by Gemalto, uses some variants of the normal EMV-CAP modes, both when it used with or without USB-connection:

- Without USB-connection, some operations correspond to standard EMV-CAP modes. For instance, the *Log on* operation uses Mode 2, and the *SecureCode* option uses Mode 1, including an amount, currency, and challenge. As opposed to the other operations, for the *SecureCode* operation the user is asked to enter the transaction details and challenge *before* entering the PIN code. The user has four options for the currency: Euros, Dollars, Pounds and other (which results in a currency code of 999).

  The other options of the device are not standard EMV-CAP modes, but a slight tweak. A challenge is sent to the card, as with Mode 3, but this challenge is not identical to the one entered by the user on the keyboard. Instead, the challenge is mangled using some unknown function inside the device. Also, the response is not obtained by applying the bitfilter to the ARQC, but it does depend on it, so the response is also mangled using some unknown function[3]

  The additional mangling of the challenge and response by unknown functions means that the e.dentifier2 is not compatible with other EMV-CAP readers. Also, it prevents the construction of a software emulation of the device, as is available for other EMV-CAP readers[4].

- When used with USB cable, again some operations on the device correspond with Mode 1 or 2. In these cases, the responses returned to the PC via the USB cable are identical to the ones that would be displayed on the device when used in unconnected mode.

  Some operations work differently, as described in detail [9]: more information is sent over the USB cable to be shown the user on the display of the device and included in the calculation of the cryptograms. This solution is marketed by Gemalto as SWYS, which stands for 'Sign What You See'. In principle such a solution is more secure than normal EMV-CAP, as the user is given more understandable information about the transaction he is approving, so phishing (by Man-in-the-Browser attacks or social engineering attacks) becomes harder. However, this option was seriously flawed, because malware on the PC could by-pass the approval of the user: the user gives his approval by physically pressing the OK button on the device, but an instruction over the USB cable could by-pass this, effectively giving malware the power to press the OK button.

3.5 Internet banking fraud in the Netherlands

In the Netherlands, internet banking fraud rose sharply to peak in 2011 and 2012, and has been drastically reduced since, as shown in Figure 3.5. The reduction in fraud is not due to stronger authentication measures, with for instance EMV-CAP readers. Instead, it seems largely due to better monitoring to spot suspicious transactions and to detect money mules (i.e. the destination bank accounts used by the criminals to receive money). Another factor in the continuing decline may well be that criminals have moved on to ransomware are a better business model.

year	fraud (in M€)
2008	2.1
2009	1.9
2010	9.8 (7100€ per incident)
2011	35 (4500€ per incident)
2012	34.8
2013	9.6
2014	4.7
2015	3.7

Figure 3.2: Internet banking fraud in the Netherlands[6].

[3]Changing the least significant bits of the ARQC does not change the response code, which is a strong indication it is used as a key in some DES operation, as these bits are used as parity bits in DES and not used in the actual cryptographic operations.

[4]E.g. from `http://sites.uclouvain.be/EMV-CAP`

[6]Source: the Dutch Banking Association (`https://nvb.nl`) and the Dutch Payment Association (`https://www.betaalvereniging.nl`.)

Chapter 4

EMV Contactless

The EMV contactless specs [20, 26–34]. are publicly available from the EMVCo website. These specs also refer to the original EMV contact specs, and build on ISO/IEC 14443 for the RFID communication.

Three books in the EMV contactless specs, Book A [20], Book B [26] and Book D [34], are common for all contactless EMV cards. For Book C, the so-called kernel specification, there are different versions for the card schemes of the EMVCo members:

- Kernel 1 for Visa and JCB [27],
- Kernel 2 for MasterCard [28],
- Kernel 3 for Visa [29],
- Kernel 4 for American Express [30],
- Kernel 5 for JCB [31],
- Kernel 6 for Discover [32], and
- Kernel 7 for UnionPay [33].

These kernel specs vary greatly in size, from only 34 to over 500 pages, which is already an indication they are very different. Note that for Visa and JCB there are two kernels; below we describe how the appropriate kernel is selected.

Many of the security concerns with contactless cards stem from the wireless nature of these cards, rather than specific features of the EMV protocols. So we discuss these weaknesses in Section 4.2 before we go on to describe the details of the EMV contactless protocols.

Section 4.3 describes the general features of EMV contactless and then describes two kernels in more detail:

- MasterCard 2 aka PayPass, in Section 4.4, and
- Visa kernel 3 aka PayWave in Section 4.5.

We only describe these two kernels for the simple reason that we only had access to these cards to analyse. The description in this chapter is largely based on [53].

Visa markets its contactless transaction system as *payWave*, a term not used in the specifications [27, 29]. MasterCard markets uses the name *PayPass*. Again, this name is not mentioned in the Kernel specifications [28]. 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 [44].

4.1 Overview of literature on EMV Contactless

Heydt-Benjamin et al. report the possibility of re-constructing mag-stripes from the data that from eavesdropped contactless communication [40].

The possibility of using NFC phones for relay attacks is first described in [37], and this set-up has subsequently been used for relaying contactless payments by various authors [43, 50, 53].

Roland and Langer demonstrated a protocol weakness in EMV contactless mag-strip mode, which allows cards to be cloned [51].

Emms et al. found a configuration flaw in dual-contact cards issued in the UK: the flaw allows cards to accept the PIN verification check via the contactless interface [19], something that should not be allowed. Students of the Radboud University found that the first dual contact cards issued by banks in the Netherlands contained similar configuration flaw.

4.2 Known attacks and weaknesses in contactless EMV cards

Many of the security concerns for contactless EMV cards are not due to specific details of the precise protocol involved, but are due to more generic and fundamental features. One feature is that the contactless nature of cards presents a possible attack vector to the attacker: the attacker can try to eavesdrop on communication or surreptitiously activate a card without the card. This is aggravated by a second feature: that transaction are typically done without the user entering a PIN.

Because of this, first we discuss the security concerns for contactless EMV before diving into the details of the protocol. Before looking at specific attacks, we first discuss the general attack vector provide by wireless nature of communication with the card.

4.2.1 Passive and active attacks on the contactless interface

One disadvantage of contactless cards is that the card holder does not have to insert his card in a reader to explicitly give consent to it being read. There are two different attack scenarios here: (1) an attacker could *passively eavesdrop* on the interaction between the card and terminal, and (2) an attacker could *actively interact* with a card without the card holder realising. The first attack introduces a risk for eavesdropping, the second is more dangerous, as it can be used for relay attacks.

Passive eavesdropping is possible at much larger distances than activation. Activation is harder at a distance, because enough power has to be delivered to the card to power it, and the power required for this increases rapidly with the distance. The largest distance for passive eavesdropping reported in the literature is 18 meters [35]. The largest distance for activation is only 50 cm [38].

Large activation distances require larger and very powerful antennas. For a relay attack an attacker is probably more likely to use readily available hardware – in particular, NFC phones [12, 37, 53] – and content himself with a much smaller range of only several centimetres, instead of using more powerful but unwieldy antennas for longer ranges described in the literature [38, 39, 42].

4.2.2 Attacks and weaknesses in contactless EMV cards

The contactless nature of cards plays is a factor in many of the security concerns with contactless EMV.

- **Cloning cards for mag-stripe operations**

  On some early RFID credit cards, it was possible to reconstruct the mag-strip using data obtained via the contactless interface [40]. Note that this is the same flaw that was in early contact EMV cards, discussed in Section 2.4, where the mag-stripe could be reconstructed from information obtained via the contact interface.

- **Privacy**

  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. On Dutch payment cards issued by banks, the PAN reveals the bank account number, which could be useful for identity theft.

  In any case, the cards always send out a fixed UID as part of the ISO 14443 anti-collison protocol, which uniquely identifies a card. The anti-collision protocol does support cards sending a random UID, which is typically used in e-passports¿

- **Cloning cards for contactless mag-stripe operations**

  Some contactless EMV cards support a contactless mag-stripe mode, which is similar to a traditional transaction using the mag-strip, but with an additional security measure that the chip provides a dynamic 3 digit code, called CVC3. This CVC3 can be used in place of the static CVC written on the back of a credit card.

  The goal of this dynamically generated code is that cards cannot be cloned, but Roland and Langer showed that it may be still be possible to clone a contactless card when this mode is used, due to problems with the unpredictable numbers [51].

- **Contactless PIN verification**

  Contactless EMV cards are often dual contact, meaning they provide both the contactless and the contact interface. On such a card, it can be configured which functionality is provided over which of the interfaces, or both. Emms et al. discovered that some UK cards were misconfigured here, in that the `VERIFY` command was available via the contactless interface, whereas it should only be available via the contact interface [19]. Students at the Radboud University discovered that contactless EMV cards issued by Dutch banks were also misconfigured in this way. Though for these cards it was not possible to perform the `VERIFY` after the EMV application was selected, this was possible if no application was selected yet.

  One possible way to abuse the contactless access to PIN verification suggested by Emms et al. [19] is to repeatedly guess the PIN over a long period. We think that a more realistic risk is a Denial of Service attack, where the attacker blocks a card by contactlessly guessing the PIN 3 times.

- **Relay attacks**

  Relay attacks have been demonstrated on EMV contactless transaction by several researchers [12, 37, 50, 53]. Phones with NFC support provide readily available hardware to carry out these attacks.

  Normally, timing is a practical complication in relay attacks: the relay has to be fast enough for the terminal not to time out. However, the time-out for contactless terminals turns out to be extremely long – above 50 seconds [53]. Chothia et al. have proposed improvements to the EMV contactless protocol to make relays harder [12], by including time-critical steps.

  In version 2.5 of MasterCard's Kernel 2 specifications, which was released in March 2015, a protocol is introduced to make it harder to perform relay attacks [28, Section 3.10]. This protocol is very similar to the one proposed in [12], except that an additional command and protocol step is introduced to enable this functionality.

  So far (in early 2016) there is no evidence that criminals see relay attacks as an interesting option. The criminal business model is not so attractive, as only small amounts can be stolen, and not as cash, and the money can be traced. This means that relay attacks do not easily scale to a interesting level. Also, criminals may run the risk of being caught if they have go around with an antenna or an NFC phone to connect with bankcards people keep in their pockets or handbags. To our knowledge, by spring 2016 only one instance of a relay attack has been reported in press[1].

[1] `http://www.scmagazineuk.com/sc-staff-hit-by-contactless-card-theft/article/447971/`

- **High value transactions in foreign currencies without PIN?** Further research by Emms et al. suggest it might be possible for an attacker to confuse a card into approving a high value foreign currency transactions without using the PIN [18]. However, it has not been shown yet whether this is possible in practice.

4.3 EMV Contactless Transactions

The EMV contactless specifications specify two modes of operations: **EMV Mode** and **Mag-Stripe Mode**. Which mode is provided by a terminal depends on the infrastructure between the terminal and the back-end. How each mode works exactly differs per kernel and is specified in the kernel's corresponding Book C.

EMV Contactless transactions are in many ways similar to EMV Contact transactions. Before diving into the details, it is useful to point out some important differences:

1. A mag-stripe mode is provided to support older infrastructure that cannot handle EMV data yet.

2. Online contactless transactions typically involve only one cryptogram, and not two, as is common in contact transactions. The check with the issuer then takes place after the card has left the proximity of the reader.

   This is presumably done to reduce the time that the card has to be held close to the reader. This reduces the risk of so-called *card tears*, where communication is interrupted because the card is moved away from the reader to soon.

3. An additional CVM method is added: *Consumer Device CVM*. This uses a mobile device of the cardholder to verify the identity of the cardholder, in case the contactless transaction is carried out with a NFC phone.

4.3.1 Kernel selection

To start a contactless transaction the terminal has to select an applet on the card. As with contact EMV, contactless cards contain a list of AIDs of applets that are present combined with a priority per applet. For contactless cards this list is specified in the *Proximity Payment System Environment* (PPSE), which is included in the file *2PAY.SYS.DDF01*. A terminal will select the applet with the highest priority that it has support for. The terminal will then use the kernel specification that corresponds to the selected AID.

For Visa and JCB's Kernel 3, Mag-Stripe Mode is preferred if both the terminal and card support it. This is specified in requirement 5.2.2.3 from the specification [29]: "*If the reader is both Mag-Stripe Mode-enabled and [contactless] EMV Mode-enabled [...] [and] If the card indicates Mag-Stripe Mode is supported [...] then the kernel shall proceed with Mag-Stripe Mode*". In fact, cards using Kernel 3 cannot indicate whether they support EMV Mode. Bit 8 of the second byte of the AIP indicates whether mag-stripe is supported [29, Annex A.2].

For Kernel 2, it is the other way around: cards cannot indicate support for Mag-Stripe Mode but only whether EMV Mode is supported. Similar as for Kernel 3, bit 8 of the second byte of the AIP is used to indicate support for EMV Mode [28, Annex A.1.16]. If EMV Mode is supported by both the terminal and card, this is the preferred method.

4.4 MasterCard PayPass (Kernel 2)

The MasterCard contactless implementation PayPass is specified in Book C-2 [28]. It supports two types of transactions: EMV Mode and Mag-Stripe Mode. Mag-Stripe Mode is very different from normal EMV transaction and performs payments based on magnetic stripe-like data obtained from the card.

Terminals can be configured to only support Mag-Stripe Mode or EMV Mode. Cards however can only indicate whether they support EMV Mode and have no indication for support for Mag-Stripe Mode.

The PayPass M/Chip requirements [44] define which modes are supported by the different cards: MasterCard-branded cards *must* support Mag-Stripe Mode and *may* support EMV Mode, Maestro-branded cards *must* support EMV Mode and *must not* support Mag-Stripe Mode. A Maestro-branded card must not support Mag-Stripe Mode but there is no option to indicate this to the reader.

4.4.1 MasterCard Contactless Transaction Initialisation

The initialisation of both EMV and Mag-Stripe Mode transaction is the similar as with contact EMV. The terminal starts a transaction by reading the PPSE, to which the card responds with the list of available payment applications with corresponding priorities. The terminal selects the application with the highest priority and then sends the `GET PROCESSING OPTIONS` command. The card responds by sending the AIP, which the terminal can use to determine the transaction type, and the AFL, which contains the list of all files the terminal can read.

The AIP on the card can indicate whether EMV Mode is supported, but not whether Mag-Stripe Mode is supported. If both the card and the terminal support EMV Mode this transaction mode is used. Otherwise Mag-Stripe Mode is used only if it is supported by the terminal. If no mode is supported by both the card and terminal, selection of the application failed and the terminal can try to select the next application.

4.4.2 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. As opposed to EMV Contact and EMV Contactless Mode, Contactless Mag-Stripe Mode does not 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 [51].

A transaction using Mag-Stripe Mode is initialised as discussed in Section 4.4.1. If the terminal decides that mag-stripe will be used, necessary data is read using `READ RECORD` commands. This data includes fields necessary to construct Track 1 and/or Track 2 data. Now, instead of the `GENERATE AC` command, as used in regular EMV transactions, the new `COMPUTE CRYPTOGRAPHIC CHECKSUM` command is sent, where the Unpredictable Number is included as a parameter. When the transaction is approved the card will return the ATC, the CVC3 (Card Validation Code) and optionally the POS Cardholder Interaction Information (PCII). The CVC3 is a cryptogram calculated over the UN used to construct the Track 1 and/or Track 2 data, that are used to authorise the transaction in legacy mag-stripe systems. It is backwards compatible with the static CVC and CVC2 on the mag-stripe and on the back of the card respectively. The PCII can be included if the transaction was performed using a mobile phone instead of a bank card. It can indicate, for example, whether an offline PIN was successful and is used to provide the terminal with the capability to present more meaningful error messages to the user.

4.4.3 MasterCard Contactless EMV Mode

Card Authentication When all the necessary files have been read during the initialisation described in Section 4.4.1, the terminal risk management determines whether card authentication should take place.

As mentioned earlier, DDA is no longer supported for contactless transactions, because, unlike SDA and CDA, DDA requires additional communication steps. SDA and CDA work as for contact transactions (described in sections 2.3.2.1 and 2.3.2.3). Older cards may support SDA but new cards must only support CDA [44].

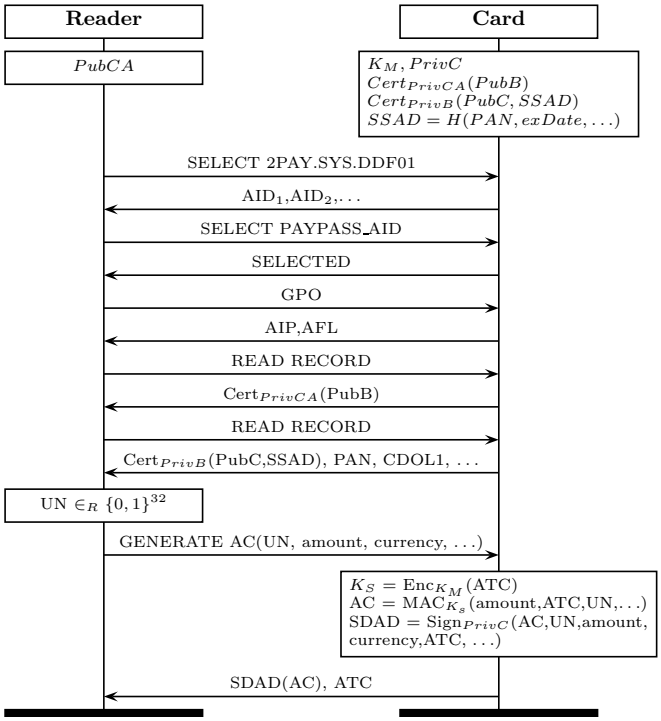

Figure 4.1: The PayPass Protocol

Cardholder Verification The terminal risk management also decides which CVM should be performed. A new cardholder verification method, called *On-Device Cardholder Verification*, is introduced. 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's device then knows On-Device Cardholder Verification must be performed and will perform the verification. How the device should verify the identity is not specified, although it is mentioned in the MasterCard Best Practices [45] 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*". The 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 authentication is not performed correctly.

If the terminal decides cardholder verification is necessary but On-Device Cardholder Verification is not supported by the card or terminal, 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 as parameters with the `GENERATE AC` command. As with EMV

Contact, the card responds with a cryptogram, though 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 may also be proof that an online transaction took place after the card received and checked online authorisation from the issuer.
- An ARQC indicates that the card or the terminal finds it necessary that the issuer authorises 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, as for EMV Contact transactions.

As in EMV Contact, when the terminal requests a TC the card can respond with a TC to approve the transaction or with an ARQC to force the transaction to be performed online. In both cases the interaction with the card is completed after the card returns a cryptogram. If an online transaction was requested the terminal will forward the ARQC, optionally with the PIN, to the issuer to get an approval or rejection for the transaction. The terminal can also decide the transaction needs to be performed online by requesting an ARQC directly. In this case the card must return an ARQC for a successful transaction. Again, the interaction between card and terminal is terminated after this and the terminal forwards the ARQC to the issuer optionally with the PIN.

The PIN is optionally included with the ARQC to the issuer depending on whether the terminal indicated to the card that 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 verification should be performed.

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

Torn Transactions The specification provides a procedure to recover torn transactions, where the communication between the card and terminal was interrupted, for example because the card was removed from the reader's field [28, Section 3.7]. Support for the recovery of torn transactions by a card is indicated by including a non-empty *Data Recovery Data Object List* (DRDOL) in its data. A terminal can keep a log of transactions that it started but were torn, i.e. no response was received to the `GENERATE AC` command. If it detects a card that occurs in the torn transaction log, it can try to recover this transaction. For this a new command, `RECOVER AC`, is introduced. The terminal will send the `RECOVER AC` command together with the data indicated in the DRDOL to the card instead of the `GENERATE AC` command. If the card previously completed the `GENERATE AC` command successfully, it will return the response to this again and the terminal can finish the transaction. If the card indicates it cannot recover the transaction, the terminal can immediately send the `GENERATE AC` command to complete the transaction.

Relay Protection To protect against relay attacks an extension to the protocol is introduced in [28, Section 3.10]. It makes use of a challenge-response mechanism, where the terminal measures the time it takes the card to reply to a command. The command used for this is `EXCHANGE RELAY RESISTANCE DATA`. The terminal will send a random number (*Terminal Relay Resistance Entropy*, which will also be used as the Unpredictable Number for the rest of the transaction). The card will respond with another random number (*Device Relay Resistance Entropy*) and three timing estimates (minimum time for processing, maximum time for processing and estimated transmission time). When the maximum time is exceeded, the terminal can retry up to two times (with a fresh random number). The Terminal Action Codes can be used to determine what needs to happen if the timings are too high (decline transaction or perform online transaction). Both random number and the timings returned by the card are included in the CDA computation. The card indicates in the Application Interchange Profile (AIP) whether it supports this relay protection. An example of the protocol including the relay protection can be found in Figure 4.2.

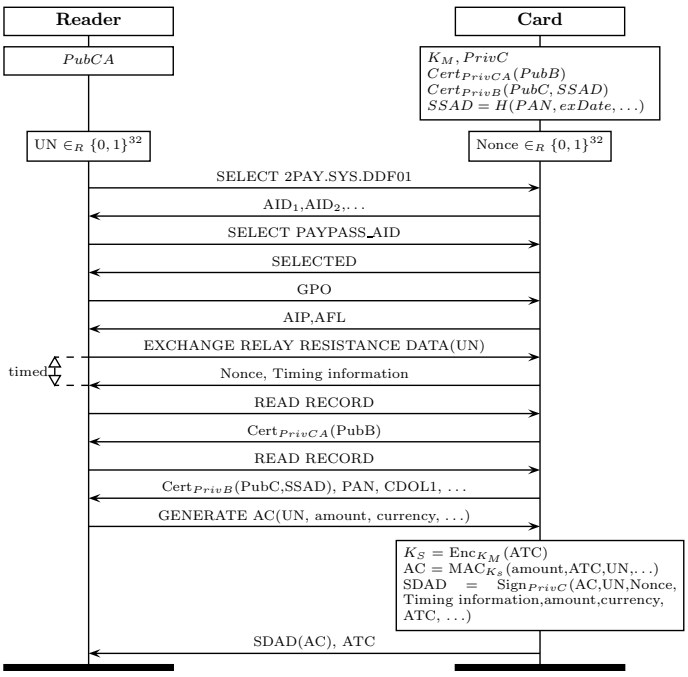

Figure 4.2: The PayPass Protocol including relay protection

Data Storage Data Storage is a new feature that allows the terminal to use a 'scratch pad' on the card and is available in two different types [28, Section 3.6]. 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 reduces the total time of the transaction as these commands will already be executed in a regular transaction anyway.

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

- *Reader Contactless Floor Limit*, indicates the transaction amount above which transactions must be authorised 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 in the terminal risk management to decide which CVM should be used and whether the transaction should be performed online.

Comparing EMV Contactless Mode with EMV Contact Below we discuss the main differences between the contactless EMV Mode and EMV Contact.

1. Card authentication methods SDA and CDA are still used, but DDA no longer is.
2. Cardholder verification through offline PIN verification is no longer considered suitable [20, Section 5.9.3].
3. Cardholder verification with a cardholder's device is supported for EMV Mode transactions.
4. The second `GENERATE AC`, needed for traditional EMV Contact transactions, is no longer supported.
5. Torn transactions can be recovered with Contactless EMV Mode. Transactions are called 'torn' when the card is removed before the communication was done.
6. A protocol is introduced to protect against relay attacks.
7. EMV Mode introduces Data Storage, with which terminals can write and read data to the card.
8. EMV Mode offers functionality for offline card balance reading.

   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.
9. 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.
10. When online PIN or signature is used as CVM, the actual verification is completed after the interaction with the card is complete. This already was the case with the signature verification since there the signature is written on the printed receipt. However, On-Device Cardholder Verification "*is completed before the interaction begins*", according to PayPass M/Chip Requirements document [44], which is counter-intuitive because before the interaction begins, the amount is not known, effectively making the verification less useful.

4.5 Visa payWave (Kernel 1 and 3)

The Visa payWave specifications are described in detail in Book C-1 and C-3 [27, 29]. As with any EMV transaction, Visa payWave transactions start with the selection of the AID by the terminal. The card's response indicates whether the selection of the application was successful. Included in this response is the card's PDOL. The PDOL consists of the data objects that the card expects as input from the terminal with the `GET PROCESSING OPTIONS` command. In addition to the `GET PROCESSING OPTIONS` command, the `EXTENDED GET PROCESSING OPTIONS` command is introduced to support Integrated Data Storage.

Depending on the response of the card and the capabilities of the terminal, the terminal chooses one of the following three contactless options:

- *Mag-Strip Mode*,
- *qVSDC Mode*, or

- *VSDC Mode.*

In case none of these are possible, the terminal falls back to using the contact chip or magnetic stripe.

The three contactless options above are listed in order of descending priority according to requirement 5.2.2.3 from the specification [29]. So for example, if a card and terminal both support qVSDC mode and Mag-Stripe Mode, the transaction will be performed with Mag-Stripe Mode. Below the three contactless options are described in more detail.

4.5.1 Contactless Mag-Stripe Mode

The terminal decides whether a transaction should be processed as Mag-Stripe Mode if the AIP, contained in the response of the card to the `GET PROCESSING OPTIONS` command, indicates that this mode should be used.

In this mode the card provides 'static' information similar to the information read from a traditional mag-stripe, i.e. the Track 2 Equivalent data and optionally Track 1 Discretionary Data and Cardholder Name. However, the card will provide some additional 'dynamic' information. The terminal can request an online cryptogram. If no cryptogram is requested, the card generates a 3 digit *dCVV (dynamic Card Verification Value)* that will be included in the Track 2 Equivalent data. This dCVV is only based on the PAN, ATC and some other data available on the card. Note that no data from the terminal is included in this computation. If a cryptogram was requested this will be included in the response to the `GET PROCESSING OPTIONS` command. Otherwise, the AFL will at least indicate the record containing the Track 2 Equivalent data, which contains the dCVV.

4.5.2 qVSDC Mode

With qVSDC, the number of commands needed to perform a transaction is reduced and the card provides the cryptogram directly with the response to the `GET PROCESSING OPTIONS` and the `GENERATE AC` command is no longer used (see Figure 4.3). The outcome of the card's risk management, together with the capabilities of the terminal, determines whether the transaction is performed either online or offline:

- If the transaction is performed online, an ARQC is returned. The terminal needs to go online to provide the ARQC to the issuer. The issuer then indicates to the terminal whether the transaction is accepted or rejected, or whether the PIN is needed.
- If the transaction is performed offline, a TC is returned.

To authenticate the data that is returned, a new data authentication mechanism called *fDDA* is used. For this a signature is computed that includes a nonce from the card (included in the *Card Authentication Related Data*), and it is returned together with the cryptogram in the response to the `GET PROCESSING OPTIONS` command. However, unlike PayPass, the signed data does not include the cryptogram. Therefore, it might be possible for an attacker to send a shop reader a valid SDAD and an invalid AC, which the shop would not discover until it sends the AC to its bank to complete the payment. The certificate for the key that is used to generate the signature and the *Card Authentication Related Data* can be retrieved using the `READ RECORD` command from files indicated in the AFL in the response to the `GET PROCESSING OPTIONS`.

4.5.3 VSDC Mode

For Visa's VCDC mode, a regular EMV transaction is performed according to the EMV Contact specifications. The only difference here is the communication medium (RFID vs contact).

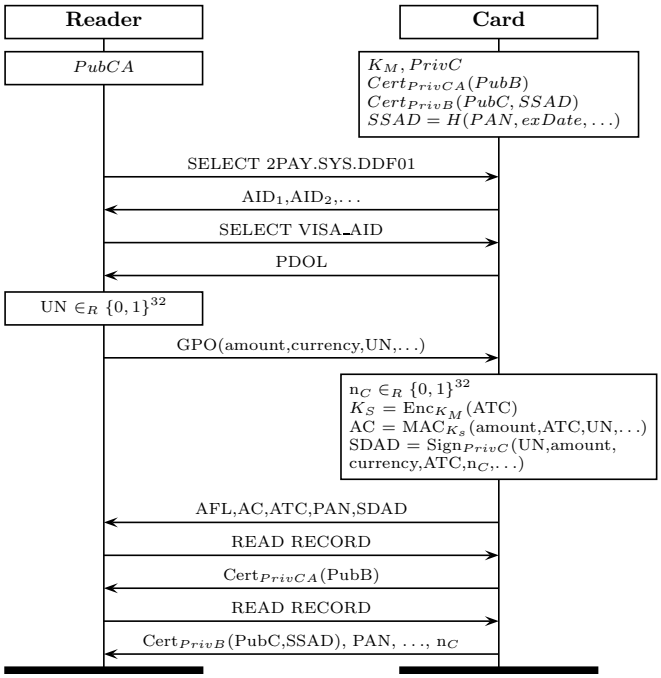

Figure 4.3: A complete qVSDC transaction

Bibliography

[1] Fides Aarts, Erik Poll, and Joeri de Ruiter. Formal models of bank cards for free. In *International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 461 – 468. IEEE, 2013.

[2] Ben Adida, Mike Bond, Jolyon Clulow, Amerson Lin, Ross Anderson, and Ronald L Rivest. On the security of the EMV secure messaging API. In *Security Protocols*, pages 147–149. Springer, 2007.

[3] Ben Adida, Mike Bond, Jolyon Clulow, Amerson Lin, Steven J. Murdoch, Ross Anderson, and Ron Rivest. Phish and chips: Traditional and new recipes for attacking EMV. In *Security Protocols*, volume 5087 of *LNCS*, pages 40–48. Springer, 2009.

[4] Ross Anderson and Steven J. Murdoch. EMV: why payment systems fail. *Communications of the ACM*, 57(6):24–28, 2014.

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

[6] Andrea Barisani, Daniele Bianco, Adam Laurie, and Zac Franken. Chip & PIN is definitely broken. Presentation at CanSecWest Applied Security Conference, Vancouver 2011. Slides available at `http://dev.inversepath.com/download/emv/emv_2011.pdf`., 2011.

[7] Karthikeyan Bhargavan, Cédric Fournet, Andrew D. Gordon, and Stephen Tse. Verified interoperable implementations of security protocols. In *Computer Security Foundations Workshop (CSFW)*, pages 139–152. IEEE, 2006.

[8] Bruno Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Computer Security Foundations Workshop (CSFW)*, pages 82–96. IEEE, 2001.

[9] Arjan Blom, Gerhard de Koning Gans, Erik Poll, Joeri de Ruiter, and Roel Verdult. Designed to fail: A USB-connected reader for online banking. In *NordSec*, volume 7616 of *LNCS*, pages 1–16. Springer, 2012.

[10] Mike Bond, Omar Choudary, Steven J. Murdoch, Sergei Skorobogatov, and Ross Anderson. Chip and skim: cloning EMV cards with the pre-play attack. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 49–64. IEEE, 2014.

[11] Check-In-Phone – Technology and Security, 2008. Available from `http://upload.rb.ru/upload/users/files/3374/check-in-phone-technologie_security-english-_2010-08-12_20.05.11.pdf`.

[12] Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel, and Matthew Thompson. Relay cost bounding for contactless EMV payments. In *Financial Cryptography and Data Security*, volume 8975 of *LNCS*, pages 189–206. Springer, 2015.

[13] Joeri de Ruiter. *Lesson learned in the analysis of the EMV and TLS security protocols*. PhD thesis, Radboud University Nijmegen, 2015.

[14] Joeri De Ruiter and Erik Poll. Formal analysis of the EMV protocol suite. In *Theory of Security and Applications (TOSCA 2011)*, volume 6993 of *LNCS*, pages 113–129. Springer, 2012.

[15] M. de Soete and M. Ward. EMV. In H.C.A. van Tilborg, editor, *Encyclopedia of cryptography and security*, pages 197–202. Springer, 2005.

[16] Saar Drimer, Steven J. Murdoch, and Ross Anderson. Thinking inside the box: system-level failures of tamper proofing. In *IEEE Symposium on Security and Privacy*, pages 281–295. IEEE, 2008.

[17] Saar Drimer, Steven J. Murdoch, and Ross Anderson. Optimised to fail: Card readers for online banking. *Financial Cryptography and Data Security*, 5628:184–200, 2009.

[18] Martin Emms, Budi Arief, Leo Freitas, Joseph Hannon, and Aad van Moorsel. Harvesting high value foreign currency transactions from EMV contactless credit cards without the pin. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 716–726. ACM, 2014.

[19] Martin Emms, Budi Arief, Nicholas Little, and Aad van Moorsel. Risks of offline verify PIN on contactless cards. In *Financial Cryptography and Data Security (FC 2013)*, volume 7859 of *LNCS*, pages 313–321. Springer, 2012.

[20] EMVCo. *Book A: Architecture and General Requirements v2.6*. March.

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

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

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

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

[25] EMVCo. *EMV Payment Tokenization Specification. Technical Framework*. 2014.

[26] EMVCo. *Book B: Entry Point Specifications v2.5*. EMV Contactless Specifications for Payment Systems. March 2015.

[27] EMVCo. *Book C-1 Kernel 1 Specification v2.5*. EMV Contactless Specifications for Payment Systems. March 2015.

[28] EMVCo. *Book C-2 Kernel 2 Specification v2.5*. EMV Contactless Specifications for Payment Systems. March 2015.

[29] EMVCo. *Book C-3 Kernel 3 Specification v2.5*. EMV Contactless Specifications for Payment Systems. March 2015.

[30] EMVCo. *Book C-4 Kernel 4 Specification v2.5*. EMV Contactless Specifications for Payment Systems. March 2015.

[31] EMVCo. *Book C-5 Kernel 5 Specification v2.5*. EMV Contactless Specifications for Payment Systems. March 2015.

[32] EMVCo. *Book C-6 Kernel 6 Specification v2.5*. EMV Contactless Specifications for Payment Systems. March 2015.

[33] EMVCo. *Book C-7 Kernel 7 Specification v2.5.* EMV Contactless Specifications for Payment Systems. March 2015.

[34] EMVCo. *Book D: Contactless Communication Protocol v2.6.* EMV Contactless Specifications for Payment Systems. March 2016.

[35] M. Engelhardt, F. Pfeiffer, K. Finkenzeller, and E. Biebl. Extending ISO/IEC 14443 Type A eavesdropping range using higher harmonics. In *Proceedings of 2013 European Conference on Smart Objects, Systems and Technologies (SmartSysTech)*, pages 1–8. IEEE, 2013.

[36] Houda Ferradi, Rémi Géraud, David Naccache, and Assia Tria. When organized crime applies academic results - a forensic analysis of an in-card listening device. *Journal of Cryptographic Engineering*, pages 1–11, 2015.

[37] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical NFC peer-to-peer relay attack using mobile phones. In SiddikaBerna Ors Yalcin, editor, *Radio Frequency Identification: Security and Privacy Issues (RFIDSec 2010)*, volume 6370 of *LNCS*, pages 35–49. Springer, 2010.

[38] René Habraken, Peter Dolron, Erik Poll, and Joeri de Ruiter. An rfid skimming gate using higher harmonics. In *RFIDsec 2015*, number 9440 in LNCS, pages 122–137. Springer, 2015.

[39] Gerhard P. Hancke. Practical eavesdropping and skimming attacks on high-frequency rfid tokens. *J. Comput. Secur.*, 19(2):259–288, 2011.

[40] Thomas S. Heydt-Benjamin, Daniel V. Bailey, Kevin Fu, Ari Juels, and Tom OHare. Vulnerabilities in first-generation rfid-enabled credit cards. In *Financial Cryptography and Data Security*, volume 4886 of *LNCS*, pages 2–14. Springer, 2007.

[41] Doc 9303 – Machine readable travel documents – Part 1–2. Technical report, ICAO, 2006. Sixth edition.

[42] Ilan Kirschenbaum and Avishai Wool. How to build a low-cost, extended-range RFID skimmer. In *Proceedings of the 15th USENIX Security Symposium*, pages 43–57. Usenix, 2006.

[43] 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.

[44] 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].

[45] 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].

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

[47] Diego Ortiz-Yepes. A review of technical approaches to realize NFC mobile payments. To appear.

[48] Diego Ortiz-Yepes. A critical review of the EMV Payment Tokenisation Specification. In *Computer Fraud and & Security*, pages 5–12. Elsevier, October 2014.

[49] Diego A. Ortiz-Yepes. Enhancing Authentication in eBanking with NFC-Enabled Mobile Phones. Master's thesis, Technische Universiteit Eindhoven, 2008.

[50] Michael Roland, Josef Langer, and Josef Scharinger. Applying relay attacks to google wallet. In *Near Field Communication (NFC), 2013 5th International Workshop on*, pages 1–6, February 2013.

[51] 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.

[52] Jean-Pierre Szikora and Philippe Teuwen. Banques en ligne: à la découverte d'EMV-CAP. *MISC (multisystem & internet security cookbook)*, 56, 2011.

[53] Jordi van den Breekel. A Security Evaluation and Proof-of-Concept Relay Attack on Dutch EMV Contactless Transactions. Master's thesis, Technische Universiteit Eindhoven, 2014.

[54] Els Van Herreweghen and Uta Wille. Risks and potentials of using EMV for internet payments. In *Proceedings of the 1st USENIX Workshop on Smartcard Technology*. USENIX Association, 1999.

[55] Thomas Weigold, Thorsten Kramp, Reto Hermann, Frank Höring, Peter Buhler, and Michael Baentsch. The Zurich Trusted Information Channel — an efficient defence against man-in-the-middle and malicious software attacks. In *Trust'08*, LNCS, pages 75–91. Springer, 2008.