SoK: Attacks on Modern Card Payments

Xenia Hofmeier[1], David Basin[1], Ralf Sasse[1], and Jorge Toro-Pozo[2]
[1] Department of Computer Science, ETH Zurich, Switzerland
{xenia.hofmeier, basin, ralf.sasse}@inf.ethz.ch
[2] SIX Digital Exchange, Switzerland
jorge.toro@sdx.com

Version 1.0
April 4, 2025

Abstract

EMV is the global standard for smart card payments and its NFC-based version, EMV contactless, is widely used, also for mobile payments. In this systematization of knowledge, we examine attacks on the EMV contactless protocol. We provide a comprehensive framework encompassing its desired security properties and adversary models. We also identify and categorize a comprehensive collection of protocol flaws and show how different subsets thereof can be combined into attacks. In addition to this systematization, we examine the underlying reasons for the many attacks against EMV and point to a better way forward.

1 Introduction

EMV, named after its founding organizations Europay, Mastercard, and Visa, is the de facto protocol standard for smart card payments. The protocol specifications are maintained by EMVCo, a consortium consisting of American Express, Discover, JCB, Mastercard, UnionPay, and Visa [22]. With 12.8 billion EMV cards, EMV transactions constitute 94% of card-present chip transactions [23]. Its central role in worldwide payments makes EMV highly security critical.

EMV consists of a family of payment technologies, and the most well known are EMV contact and contactless. In a contact transaction, the card is inserted into a payment terminal, whereas contactless cards communicate with terminals wirelessly over NFC. Mobile phone-based card payments, like Apple Pay, Samsung Pay, and Google Pay, also use this protocol over the phone's NFC interface. The contactless protocol is based on the older contact protocol and there are eight so-called kernels, which are variants of the contactless protocol, each associated with a different EMVCo member. The protocol is highly complex given the eight kernels, numerous configuration options, and a specification of over 2500 pages, spanning 15 books with many cross-references.

The current chip-based payment cards' predecessors, magnetic stripe cards, used just static data for payments, making them susceptible to simple attacks such as cloning attacks. The integrated circuits of EMV contact cards prevent such attacks with dynamically generated data in each transaction. However, EMV contact is still vulnerable to attacks, the most prominent being the machine-in-the-middle (MITM) attack that enables the adversary to pay with a card while providing the wrong PIN [35]. However, the attack's bulky, wired MITM infrastructure is difficult to conceal and makes the attack impractical. In contrast, the various attacks on EMV contactless target the contactless cards' wireless interface, which can be easily and discretely accessed by the attacker. The introduction of phone-based payments also allows for an inconspicuous implementation of a card emulator where, for store employees, a normal phone payment is indistinguishable from a malicious emulation on a smartphone. This makes attacks on EMV contactless both inconspicuous and easy to carry out.

We focus in this SoK on modern payment cards. Thus, we focus on chip-based payment cards rather than older technologies like magnetic stripes. Moreover, from all the chip-based standards, we focus on EMV, as it specifies the most prevalent chip transactions. We further restrict our scope to EMV contactless as it is the newest standard, has a large attack surface, and is still evolving. Moreover, the payment industry's push for EMV contactless has led to its wide adoption. Thus, attacks on EMV contactless are highly relevant.

Note that payment cards are used in a wide range of transaction types. Most payment cards have interfaces for contactless and contact EMV as well as magnetic stripe transactions. The information printed on the card can be used for card-not-present transactions, for example over the Internet or phone. And cards can be used at ATMs or as a second factor for e-banking. In this SoK, we focus on attacks targeting EMV contactless transactions, and we therefore do not consider EMV technologies [21] such as EMV contact, 3-D secure, and Payment Tokenisation. We also focus on attacks targeting protocol weaknesses. Thus, we do not consider social engineering attacks, ATM skimmers, or weaknesses in the bank's processes, such as weak second factor authentication or problematic card lock policies as presented in [2].

Contribution In this SoK, we summarize and categorize attacks on EMV contactless. We describe this protocol, its security requirements, relevant adversary models, and its flaws.

We provide a comprehensive framework to evaluate the security of EMV contactless consisting of security properties and adversary models. This is important to determine the practicality and relevance of attacks, i.e., whether they violate relevant properties and depend on realistic adversaries.

By extensively categorizing flaws and attacks on EMV contactless, we gain an understanding of the flaws discovered and how they are composed into different attacks. This comprehensive list helps identifying new flaws and attacks, understanding how they arise, and how we can improve the standard going forward.

$$\begin{tikzcd}\text{Card} \arrow[r, leftrightarrow, "\substack{\text{NFC}\\\text{Channel}}"] & \text{Terminal} \arrow[r, leftrightarrow, "\substack{\text{Acquirer}\\\text{Network}}"] & \text{Acquirer} \arrow[r, leftrightarrow, "\substack{\text{Payment}\\\text{Network}}"] & \text{Issuer}\end{tikzcd}$$

Figure 1: System Model.

By studying past flaws and exploits, we also gain an understanding of their underlying causes. Examples include the trade-off between preventing attacks and having high acceptance rates, and the need for backward compatibility. We compare the approaches used to find attacks and explain the importance of both formal analysis and testing on live systems.

Outline In §2, we provide background on EMV contactless. In §3, we identify security properties that are desirable for EMV contactless. In §4, we present a family of adversary models and explain how the adversary capabilities are used in practice. In §5, we categorize the flaws that enable the attacks described in §6. In §7, we discuss the identified flaws, their possible causes, and the attack discovery process, and make suggestions for the future. In §8, we draw conclusions.

2 EMV Contactless Protocol

In this section, we summarize the EMV contactless protocol [25]. We simplify some aspects, focusing on the features that are relevant for the attacks discussed in this SoK. We first describe a general EMV contactless transaction. Afterwards we highlight some features of different versions: the Mastercard kernel, Visa kernel, and transactions with mobile devices.

2.1 Protocol Overview

An EMV transaction allows a cardholder to pay merchants. To do this, the cardholder presents their card to the merchant's payment terminal.[1] Figure 1 depicts the parties involved in an EMV contactless transaction and their communication channels. The protocol specifies which messages are sent between the *card* and the *terminal* and between the terminal and the financial institution that issued the card, called the issuing bank or just *issuer*. Two additional parties are involved in the communication between the terminal and the issuer: the financial institution that processes card payments on behalf of the merchant, called the acquiring bank or just *acquirer*, and the *payment network*, e.g. Visa or Mastercard, which connects the acquirer and issuer [5].

The EMV contactless protocol serves two main purposes. First, the issuer should be convinced that the cardholder agreed to the transaction so that the

[1]We use the term "terminal" for the entire Point-of-Sales (POS) System, consisting of the reader and terminal, as described in [25].

issuer can transfer the funds to the acquirer, and ultimately the merchant. Second, the merchant should be convinced that the presented card is legitimate and that the transaction was successful so that the merchant can accept the payment. These two purposes are achieved using two different cryptographic mechanisms. The card convinces the issuer using a MAC. This MAC is based on a symmetric key shared between the card and the issuer that is set up during the issuing process. The merchant is convinced by a signature, produced with the card's private key. The card's public key is authenticated using a global Public Key Infrastructure (PKI). We call the card's authentication to the issuer Online Transaction Authorization (OTA), as it requires the terminal to go *online* to send the MAC to the issuer, and the authentication to the terminal Offline Data Authentication (ODA), as the terminal can verify the signature *offline*.

After verifying the signature, the terminal can either decline the transaction, accept it offline or send the transaction including the MAC to the issuer for online authorization. In case of offline acceptance, the transaction data and MAC are sent later to the issuer. Thus, the terminal accepts a transaction before the issuer could verify the MAC. So, by accepting the transaction offline, the merchant risks that the transaction is later declined by the issuer, see §6.8. Offline accepted transactions can be useful in situations with limited internet access.

EMV contactless consists of eight protocol variants, called kernels. Kernels 2 - 7 are for the six members of EMVCo and kernel 1 is no longer in use. The new kernel 8 is intended to unify all existing kernels in the future.

There are two EMV contactless modes: EMV mode and mag-stripe mode. EMV mode is similar to EMV contact and mag-stripe mode transmits data similar to the data on magnetic stripes. However, mag-stripe mode is not to be confused with the physical magnetic stripes on payment cards. Mag-stripe mode's data merely resembles the data on the magnetic stripe. It is a mode of EMV contactless and thus operates over NFC. As EMV mode is more modern, we focus here on EMV mode and present mag-stripe mode in Appendix §B.1.

2.2 Protocol Phases

We split the protocol into six phases, presented in Figure 2. These phases are usually executed in this sequence but might overlap in practice. In the Select phase, the card and the terminal agree on a kernel that they subsequently run. In the Get Processing Options phase, the terminal sends transaction-specific data and the card sends its capabilities. In the Read Record phase, the card sends its static data. The card authenticates to the terminal with Offline Data Authentication (ODA), and the card authenticates to the issuer with Online Transaction Authorization (OTA). The cardholder's identity is verified using a Cardholder Verification Method (CVM), such as a PIN. We now provide more details on the protocol phases. Each phase consists of terminal commands followed by card responses. We highlight data using **bold font**.[2]

[2]Unfortunately the EMV standard contains many acronyms. We have tried to minimize their usage, but stick to the essential ones to be faithful to the standard and to set the scene for analyzing attacks.

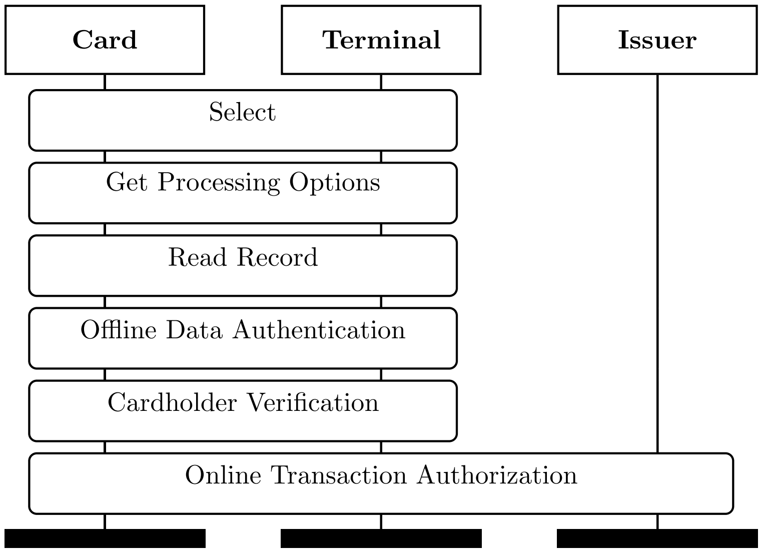


Figure 2: Phases of an EMV contactless transaction.

2.2.1 **Select Phase** (Figure 3)

The terminal and card agree on an application to run, called "Combination" in [25]. The application identifies the kernel to be run and the configuration data to be used. The terminal requests the applications that the card supports by sending a SELECT command. The card answers with a list of supported applications. The terminal chooses one of the applications and replies with a SELECT **AID** command, where the Application Identifier (**AID**) identifies the chosen application.

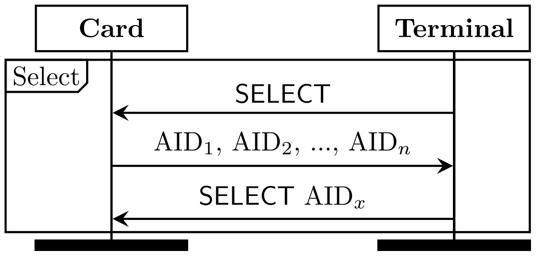


Figure 3: Select phase.

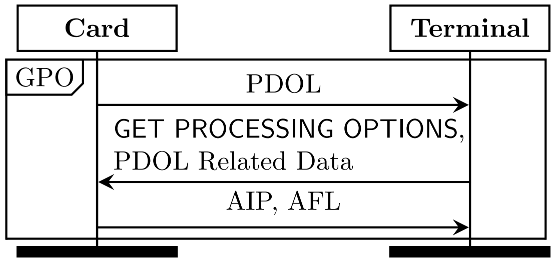


Figure 4: Get Processing Options phase

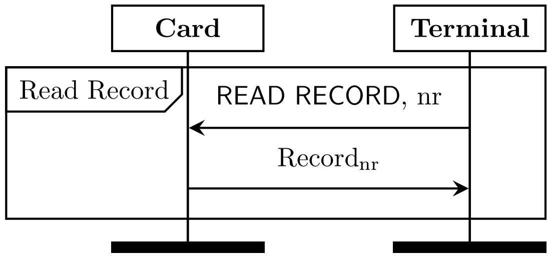


Figure 5: Read Record phase

2.2.2 Get Processing Options (Figure 4)

The card replies to the SELECT command with the Processing Data Object List (**PDOL**). In general, Data Object Lists (DOLs) specify the data elements that the card requests. The terminal provides this requested data, the **PDOL Related Data**, with the GET PROCESSING OPTIONS command. It contains transaction data such as the amount, and a terminal-generated nonce $\mathbf{UN_T}$. The card's reply contains the Application Interchange Profile (**AIP**) and Application File Locator (**AFL**). The **AIP** indicates the card's capabilities. The **AFL** identifies the files that contain the card's static data.

2.2.3 Read Record (Figure 5)

The terminal requests the card's static data, identified by the **AFL**. The terminal sends a READ RECORD command with a record number and the card responds with the requested record. The terminal repeats this for each record in the **AFL**. The terminal usually requests the card's **PAN** (i.e. the card number), expiration date, and optionally the card's public key and certificates, the Card Risk Management Data Object List (**CDOL**), and the **CVM List**. With the **CDOL**, the card requests the **CDOL Related Data**, which contains transaction data, and the **CVM List** indicates the card's supported CVMs.

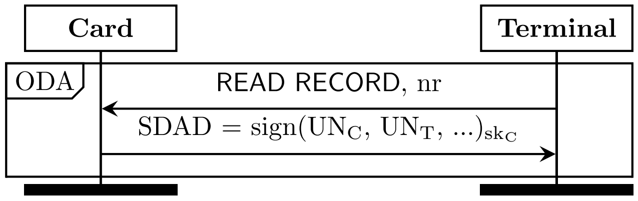


Figure 6: Overview of the ODA method fDDA.

2.2.4 Offline Data Authentication

The card authenticates to the terminal in the Offline Data Authentication (ODA) phase using the card-generated signature *Signed Dynamic Authentication Data (**SDAD**)*. Performing ODA is not required for all transactions and by all kernels, and some might require different ODA methods.

To create the signature, the card has an asymmetric key pair. The public key is authenticated using a global PKI, where root certificate authorities provide certificates to the issuers, who create certificates for their issued cards. The card sends during the Read Record phase its certificate, the issuer's certificate, and a pointer to the root certificate, called the **CA Public Key Index**. The terminal uses this pointer to look up the root CA's certificate in its database to verify the card's signature and certificates. Note that the terminal relies solely on the **CA Public Key Index** to look up the **CA Public Key**. If the card offers an invalid index, the terminal fails to select a root CA.

The most basic ODA method is Static Data Authentication (SDA), which relies on the issuer's signature on static data. Alternatively, Dynamic Data Authentication (DDA) relies on the card's signature on dynamic data, such as the terminal- and card-sourced nonces $\mathbf{UN_T}$ and $\mathbf{UN_C}$. The DDA's **SDAD** does not authenticate transaction details. In contrast, Combined DDA/Application Cryptogram (CDA) also authenticates transaction details. We now highlight the ODA methods used by Visa and Mastercard, namely fDDA and CDA.

Visa's fDDA (Figure 6). Visa's fast Dynamic Data Authentication (fDDA) is based on DDA and supports high transaction speeds. fDDA's **SDAD** contains the card's and terminal's nonces $\mathbf{UN_C}$ and $\mathbf{UN_T}$ as well as transaction and control data. The card sends the **SDAD** as READ RECORD response.

Mastercard's CDA (Figure 7). In contrast to fDDA, CDA additionally authenticates the card's MAC **AC**. The **SDAD** is sent as GENERATE AC response, see §2.2.6.

2.2.5 Cardholder Verification

The cardholder's presence is verified using one of the Cardholder Verification Methods (CVMs): Paper Signature, Online PIN, no CVM, and Consumer Device CVM (CDCVM). For Paper Signature, the cardholder signs the receipt, either on paper or on a touch screen. The cashier verifies the signature against

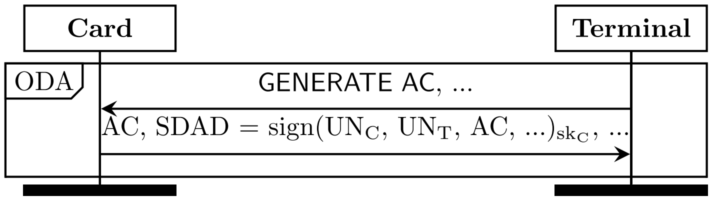


Figure 7: Overview of the ODA method CDA.

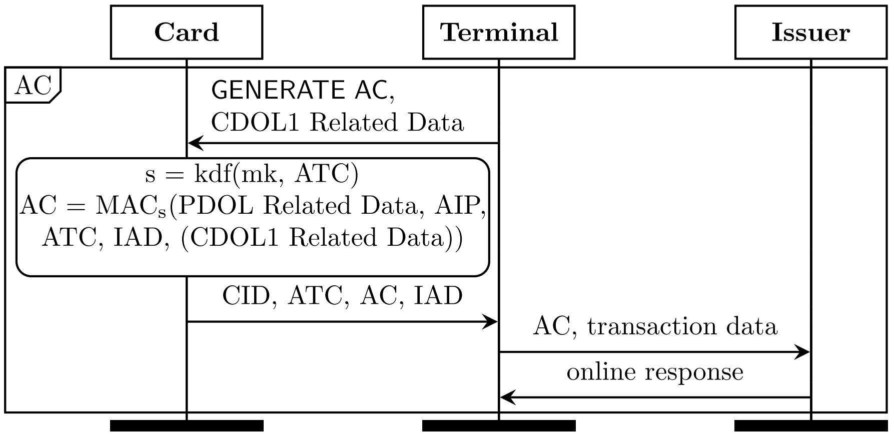


Figure 8: Overview of the Online Transaction Authorization phase.

the one on the card. For Online PIN, the cardholder enters a PIN into the terminal, which encrypts the PIN and sends it to the issuer for verification. CDCVM is performed by mobile devices such as smartphones by, e.g., verifying a PIN, a fingerprint, or a face scan.

Contactless transactions permit the use of no CVM for *low-value* transactions below a given limit. *High-value* transactions above this limit require some CVM. There can also be a limit above which contactless transactions are forbidden and a contact transaction is required. These limits vary depending on the country and payment network, and change over time. Visa and Mastercard use different mechanisms to decide which CVM is chosen, see §2.3.1 and §2.3.2.

2.2.6 **Online Transaction Authorization** (Figure 8)

The card authenticates to the issuer using a MAC, called Application Cryptogram (**AC**). The card sends it as a response to GENERATE AC and the terminal forwards it to the issuer.

The card uses a symmetric “master key” **mk**, shared between the card and the issuer, and the Application Transaction Counter (**ATC**) to derive a session key. The **ATC** is incremented for each transaction and it is authenticated by the **AC** and depending on the application also by the signature **SDAD**, see §2.2.4. Using this session key, the card computes the MAC **AC** over the transaction data. Note that the terminal does not have access to the key **mk** and thus cannot check the **AC**.

A transaction can be declined or authorized either offline by the terminal or online by the issuer. Offline authorization is not to be confused with *Offline Data Authentication (ODA)*, described in §2.2.4. The terminal requests online authorization by sending the transaction data and the **AC**. The issuer decides whether to authorize a transaction based on this data using proprietary fraud detection mechanisms [7].

2.3 Visa and Mastercard

We now highlight features of Mastercard's kernel 2 and Visa's kernel 3 and explain how they differ compared to the above general protocol phases. We focus on these kernels as they are the targets of most reported attacks.

2.3.1 Mastercard (Appendix A, Figure 9)

The Mastercard kernel (kernel 2) mostly resembles the generic phases described in §2.2. It selects the CVM according to the **AIP**, the **CVM List**, and the kernel configurations. The terminal informs the card of the chosen CVM in the **CVM Results**. First, the terminal checks if CDCVM can be performed, by checking the **AIP** and kernel configuration. If they indicate that both the card and terminal support CDCVM, the terminal checks if the transaction amount is above the CVM required limit. For low-value transactions, the **CVM Results** indicate that no CVM was performed and for high-value transactions the **CVM Results** indicate that CDCVM was performed. If CDCVM is not supported, the terminal checks if the **AIP** indicates that cardholder verification is supported. If it is not supported, the **CVM Results** indicate that no CVM was performed. Otherwise, the CVM is chosen according to the **CVM List**, which indicates the card's supported CVMs and their priorities. The terminal includes the **CVM Results** in the **CDOL1 Related Data** in the GENERATE AC command.

Kernel 2's Relay Resistance Protocol (RRP) protects against relay attacks. The card and terminal exchange timing data and nonces, which are included in the **SDAD** signature. RRP seems to be not yet available in cards or terminals [36].

2.3.2 Visa (Appendix A, Figure 10)

The Visa kernel (kernel 3) is optimized to transmit fewer command/response pairs such that the card can be quickly removed from the terminal. Thus, the Visa kernel omits the GENERATE AC command and sends the MAC **AC** in the GET PROCESSING OPTIONS response. Visa also usually performs either fDDA or online authorization, but rarely both.

Kernel 3 uses the Terminal Transaction Qualifiers (**TTQ**) and Card Transaction Qualifiers (**CTQ**) to choose a CVM. In particular, **TTQ** indicates to the card the supported CVMs and whether a CVM is required. The **CTQ** indicates to the terminal which CVMs the card requires or if it performed CDCVM.

2.4 Mobile Devices

Mobile devices differ from physical, plastic cards as they can perform cardholder verification, called CDCVM. This is done, for example, using the cardholder's PIN, fingerprint, or face scan. Apple Pay always requires CDCVM, while Google Pay allows for low-value transactions without CDCVM.

Note that Visa and Mastercard differ in how the terminal chooses CDCVM: Mastercard uses the **AIP**, while Visa uses the the **CTQ**. [36] observed that for Visa and Mastercard, the proprietary Issuer Application Data (**IAD**) indicates whether CDCVM was performed and if a plastic card or a phone performed the transaction. The **IAD** is authenticated by the MAC **AC**, which Mastercard checks, whereas Visa does not.

Transit Mode Apple Pay's "Express Transit" and Samsung Pay's "Transport card" are public transit features, called *Transit Mode* by [36]. Transit mode allows cardholders to present their phone at an origin and destination station to a dedicated transit terminal. The transit providers can thereby record the route taken and compute and charge their fee. The transit transaction is initially of value zero (and the actual cost is charged later) and for convenience it does not require a CVM. Google Pay does not offer a dedicated transit mode as it already allows for low-value transactions without CVM.

3 Security Properties

In this section, we describe the security properties that EMV should ideally achieve. We derive these properties from the stakeholder's requirements, which we list first. We then formulate these requirements more precisely as system properties. We will list for each attack the violated properties in §6 and in Table 1, in Appendix C. Note that most of the listed publications do not state which properties their attacks violate.

3.1 Stakeholder Requirements

Clearly, no stakeholder wants to lose money. Some stakeholders have additional requirements. In more detail, the cardholder requires the payment system to be available to pay with their card. The cardholder does not want to lose money from unintended payments, so only intended transactions should be performed. Finally, cardholders might desire privacy protection as card transactions can reveal personal data.

Merchants rely on the availability of the payment system. Moreover, when a transaction is accepted by the terminal, the payment should succeed at the bank so that the merchant receives its payment and for the correct amount.

The acquirer, payment network, and issuer's business involves providing a secure, available payment system to the cardholders and merchants, keeping their trust by fulfilling their requirements. This in turn entails business decisions and risk analysis. For example, it may be advantageous to incorrectly accept some payments when this reduces the chance of rejecting correct transactions. Although wrongly accepted transactions can lead to financial losses, rejected transactions can drive away customers and thereby cause even greater losses due to reduced transaction volume.

3.2 System Properties

We now formulate the above abstract stakeholder requirements as system properties. Note that some of these requirements go beyond the domain of EMV. For example, the payment transfer from the cardholder's bank account to the merchant's account is out of EMV's scope. Three of these system properties, namely P1) *data authentication*, P2) *no delayed decline*, and P5) *secrecy* were formalized by [6, 5]. We derive the other properties from the above stakeholder requirements.

P1) Data Authentication The EMV protocol should ensure that payments are performed as intended: the correct funds should be transferred from the right cardholder account to the right merchant account. This constitutes an agreement property between the card, terminal, and issuer: They all should agree on the card's **PAN** and the amount and currency of the transaction. The agents might also want to agree on additional data to rule out unintended protocol flows, like replay attacks which could lead to unintended (repeated) transactions.

P2) No Delayed Decline The issuer should not decline transactions that were accepted by the terminal. Otherwise, a terminal could accept a transaction that might later be declined by the issuer, while the customer is long gone with the goods and the merchant would not get paid for these goods.

P3) Cardholder Intent Only transactions intended by the cardholder should be performed. This is an abstract property as intent itself is difficult to formalize. Cardholder intent is approximated by two properties:

P3.1) High-Value Transactions Require CVM. CVMs increase confidence in the cardholder's intent. Thus, a CVM should be performed for high-value transactions.

P3.2) Card Close to Terminal. The card's proximity to the terminal can also imply cardholder intent. Provided the card was not stolen, we can assume that the cardholder would only put the card close to the terminal if they intend to perform a payment. However, communication over NFC can be relayed and EMV contactless currently does not provide effective counter measures against it, see §4.2 and §5.1.1. Thus, this property is violated by standard MITM attacks. In fact, such MITM attacks are an essential building block for many of the presented attacks. Thus, we do not state which attacks violate this property in §6, but list them in Table 1 in Appendix C.

P4) Privacy The cardholder might desire privacy regarding personal data and metadata that might reveal their habits. Previous versions of the EMV protocol did not have any privacy protection mechanisms in place. The new kernel 8 and the new Book E [25] introduce such mechanisms.

P5) Secrecy Some of the EMV protocol's data should remain secret to prevent fraudulent EMV contactless transactions. This data includes the card's key material and PIN. Other data can be misused in other ways such as forging magnetic stripe cards, see §5.1.12, or in card-not-present transactions. This data includes the **PAN**, expiry date, and Card Security Code (**CSC**). Note that the **CSC** is printed on the card and it is called by VISA the Card Verification Value (CVV) and by Mastercard the Card Verification Code (CVC).

P6) Availability All parties require the system's availability.

4 Adversary Model

We next present a family of adversary models under which EMV's security can be studied. We first present eight adversary capabilities and then highlight those seen in practice.

This SoK focuses on the EMV contactless protocol. Thus, we only consider adversary capabilities that target the protocol and its participants directly. We do not consider attack surfaces outside of EMV contactless, such as the magnetic stripe or the banking system and we do not consider techniques not involving the protocol such as social engineering or ATM skimming.

4.1 Adversary Capabilities

We consider the four protocol participants: the card, terminal, acquirer, and issuer, who communicate over three channels:
NFC channel: between the card and the terminal;
Acquirer network: between the terminal and the acquirer;
Payment network: between the acquirer and the issuer;

We consider three adversary capabilities. (1) *Network capabilities*: The adversary can control some of the three channels. (2) *Compromising capabilities*: The adversary can compromise some of the agents, thereby learning their secret keys. This lets the adversary impersonate this agent. (3) *Visual channel capabilities*: The adversary can visually inspect a card. This results in the following eight combinations:

A1) control NFC channel,
A2) control acquirer netw.,
A3) control payment netw.,
A4) compromise card,
A5) compromise terminal,
A6) compromise acquirer,
A7) comp. issuer, and
A8) access visual channel.

Any subset of these combinations yields an adversary model.

4.2 Adversary in Practice

We assess the adversary capabilities as seen in practice.

A1) NFC Channel The NFC interface of EMV contactless is more easily accessed by the adversary than the card-terminal channel of EMV contact. The NFC channel can be accessed from a short distance without direct contact to the card or terminal. The adversary can eavesdrop on the communication on the NFC channel. The adversary can also communicate with the card or terminal by emulating a card or terminal.

NFC's limited range can be extended by relay attacks, which have been demonstrated on EMV contactless. The relay infrastructure usually consists of two devices communicating over a relay channel. One device, the terminal emulator, communicates to the card and the other, the card emulator, communicates to the terminal. This infrastructure can be implemented using two NFC enabled Android phones [27, 17, 11] or specialized hardware [39]. A MITM attack can be performed by modifying the relayed messages. Note that a relay attack can be seen as an attack violating the card's proximity to the terminal (P3.2) rather than an adversary capability. We categorize it as an adversary capability, as it is a central building block of most of the attacks described in this SoK.

A2, A3) Acquirer and Payment Networks The EMV specification [26] specifies the acquirer interface. However, communication between the acquirer and the issuer is not part of this specification. We are unaware of practical attacks targeting the acquirer or payment networks.

A4) Compromise Card The extraction of key material through side channels, such as timing attacks [31, 14] or power analysis attacks [33], or using invasive methods such as microprobing [32] and its countermeasures are widely studied and still an active research area. Smart card manufacturers continue to deploy new countermeasures, while new attacks are being developed. Extracting the card's key material allows an adversary to impersonate the card by forging the MACs and signatures. This allows the adversary to forge transactions. As the adversary requires access to the card to extract the key materials, the benefit is unclear of extracting the key material over just using a stolen card.

A5) Compromise Terminal EMV does not specify any of the terminal's secret key material. It might however possess keys to communicate with the acquirer. In addition to revealing key material, a terminal could be modified. The viability of such modifications was demonstrated on a live system by [29], who modified the terminal's nonce choice, see §6.2.2.

The attack by [18], see §6.2.1, relies on a rogue terminal that forwards a recorded transaction. [18] implemented a proof-of-concept rogue terminal. They did not test it on a live system and thus it is unclear how practical this attack is.

Other terminal modifications include displaying a wrong payment amount to overcharge the cardholder, or recoding the entered PIN. The feasibility of arbitrary modifications was demonstrated by [15]. Such modifications would most likely need to be performed by the merchant, as the terminal providers employ physical tamper resilience. Note that if such modifications are detected, the merchant owning this terminal is not anonymous as the merchant has an account with the acquiring bank. This makes such attacks unattractive.

Note that a *fake* terminal relaying a transaction to a *real* terminal does not constitute a compromised terminal but is part of a MITM infrastructure on the NFC channel (A1).

A6, A7) Compromise Acquirer or Issuer The acquirer and issuer have control over the merchant's and cardholder's accounts respectively. Thus, an attacker controlling an acquirer or issuer can perform more powerful attacks by accessing these accounts instead of attacking the EMV protocol.

A8) Visual Adversary An adversary may visually inspect a card. Whenever a plastic card is used for a payment, it is exposed and an adversary can potentially see its details. The installation of inconspicuous cameras near terminals, as proposed by [20], is also possible. The adversary could also observe the cardholder entering the PIN into a terminal.

5 Flaws

In this section, we list the flaws identified in the EMV contactless protocol. As we will see in §6, some flaws are combined into practical attacks. In §7.1 we will discuss the flaws listed in here and highlight some common traits.

We categorize each flaw with respect to the protocol variant where it appears, namely the Mastercard or Visa kernel, or Mobile Devices, and when the flaw is not specific to a protocol variant, we categorize it under "General EMV". We list flaws on mag-stripe mode in Appendix §B.2.

5.1 General EMV

We now list the flaws not specific to any protocol variant.

5.1.1 No Relay Protection

NFC communication can be relayed, which is exploited by MITM attacks. Multiple protocols to provide relay protection have been proposed [16, 36, 11], also by Visa [10] and Mastercard, see §2.3.1. However, to our knowledge, these protocols are not yet available in cards or terminals. Moreover, [36] demonstrated the ineffectiveness of these protocols.

5.1.2 Visa Responses Built from Mastercard Responses

A terminal communicating with a Visa card should run the Visa kernel and a terminal communicating with a Mastercard card should run the Mastercard kernel. However, [5] showed that responses for the Visa kernel can be built from the responses of a Mastercard card. This enabled the *Card Brand Mixup* attack by [5], see §6.6.1. This attack is possible for the insecure Visa configuration without fDDA. Whereas transactions with fDDA prevent this attack, as Visa's **SDAD** has a header different than Mastercard's. A Mastercard **SDAD** will therefore not be accepted by the Visa kernel.

5.1.3 Not Checked if AID and PAN Match

Terminals communicating with a card should be running a kernel corresponding to the card's brand. The card's brand can be determined from the **PAN**, and the **AID** indicates which kernel the terminal is running. Thus, issuers could verify that the **PAN** and **AID** match. However, experiments by [5] and their disclosure process with Mastercard revealed that not all issuers detect mismatches. As a result of this process, Mastercard implemented those checks on their network. Note that the behavior of the issuers is not publicly specified.

5.1.4 TAC-Denial Set to Zero

The Terminal Action Code-Denial (**TAC-Denial**) specifies the acquirer's conditions that cause transactions to be declined offline. The terminal checks the **TAC-Denial** and **IAC-Denial** against the Terminal Verification Results (**TVR**) to determine if a transaction should be declined. The specification recommends not to set some Terminal Action Codes to zero.[3] However, there are no recommendations for the **TAC-Denial**. The experiments conducted by [7] showed that many terminals set the 'CDA Failed' bit in the **TAC-Denial** to zero. This prevents transactions with this bit set from being declined, which allows fraudulent transactions to be performed, such as the PIN-Bypass attack by [7], see §6.6.2.

[3]According to [26] Book 3, page 123, specific bits, including the 'CDA Failed' bit, of the **TAC-Online** and **TAC-Default** should be set to one.

5.1.5 Unencrypted Data

Some data is transmitted unencrypted in EMV transactions, and can be misused by the adversary. For example, the *Magnetic Stripe Cloning* attack [28] in §6.1.1 exploits the unencrypted **Track 2 Equivalent Data** and the *Eavesdropping on Card Data* attack [20] in §B.3.1 harvests the unencrypted **PAN**, cardholder name, and expiry date.

5.1.6 SDA and DDA do Not Authenticate the AC

ODA authenticates the card to the terminal. The ODA methods SDA and DDA only authenticate card data but not transaction data such as the MAC **AC**. [6] showed that the adversary can modify the **AC**, which results in the *Merchant-Holding-the-Bag* attack described in §6.8.

5.1.7 Weakness of Paper Signatures

Paper signatures can be forged and cashiers might not properly check signatures. Using paper signatures is even worse when the transaction is performed with a smart device like a smartphone. The cashier will use a signature presented on the phone's screen for verification. When the phone is under the adversary's control, it can show any signature. This is exploited in the *Inducing Authentication Failure* attack by [7], see §6.6.2.

5.1.8 Out-Of-Order ATCs Accepted

The **ATC** should prevent the authorization of out-of-order transactions. However, [29] showed that many issuers do not properly check the **ATC**. Their experiments showed that transactions were accepted with an out-of-order **ATC**. These checks are not part of the EMV specification.

5.1.9 UN Reuse Not Prevented

The terminal-sourced nonce $\mathbf{UN_T}$ should prevent replay attacks, but [29] showed that terminals can resend the $\mathbf{UN_T}$, enabling the *Replay with Nonce Reuse* attack, see §6.2.2.

5.1.10 AID is Not Cryptographically Protected

The card informs the terminal of its supported kernels by sending **AID**s and the terminal chooses a kernel by sending the corresponding **AID**. Neither the **AID**s sent by the card nor the terminal are cryptographically protected and hence they can be modified by an attacker. This enables the *Card Brand Mixup* attack by [5] and the downgrade attack by [7], see §6.6.1 and §6.9.1.

5.1.11 Not Checked if Plastic Cards Perform CDCVM

CDCVM can only be performed by mobile devices and therefore not by plastic cards. However, according to [29], the card's type and the performed CVM are not checked. Namely, declining transactions with CDCVM and an **IAD** indicating a plastic card, could prevent the PIN-bypass attacks by [29, 6, 5], see §6.5.1, §6.5.2, and §6.6.1.

5.1.12 Magnetic Stripe Data Present in EMV

Magnetic stripe payment cards mostly consist of two data tracks, containing persistent data such as the **PAN** and the expiration date. [28] demonstrate that these tracks can be constructed from EMV contactless transactions. The EMV **Track 1** and **Track 2 Equivalent Data** are almost identical to the magnetic stripe tracks. It can be accessed by the READ RECORD command in EMV mode and is still present in the latest kernel specifications and the new kernel 8.

The magnetic stripe tracks contain a static Card Security Code (**CSC**), which authenticates the card. Compared to this static **CSC**, the **Track 2 Equivalent Data** in EMV contains a dynamic **CSC**. This, and the fact that the discretionary data should be unique for each transaction mode, should prevent such a cloning attack. However, experiments by [28] show that these measures are not properly implemented.

5.1.13 Magnetic Stripes Are Still Supported

Most EMV cards still contain a magnetic stripe that is vulnerable to cloning. The data can be read with a publicly available reader and written to a new magnetic stripe card. This makes the magnetic stripe the weakest link on the card. Mastercard announced that they would remove magnetic stripes from future cards [30], but they are still there in 2025. Although this is not directly a flaw of the EMV contactless protocol, it makes attacks where data is extracted from an EMV transaction impactful, see §5.1.12 and §6.1.1.

5.2 Mastercard Kernel

We next list the flaw discovered in the Mastercard kernel.

5.2.1 CA Look up Failure Not Declined

One would expect a terminal to decline transactions when **CA Public Key** lookup fails. However, [7] showed that the Mastercard kernel does not always decline such transactions.

According to the Mastercard kernel specification [25], the terminal sets the 'CDA Failed' bit of the **TVR** when the **CA Public Key Index** is not present in the terminal's database. This bit is set before the GENERATE AC command and it directly affects whether CDA is performed. Namely, if the **TVR** has the ‘CDA Failed’ bit set and the **AIP** indicates that the card does not support CDCVM then the terminal does not request the **SDAD** in the GENERATE AC command. Thus, no ODA will be performed and all data that is only protected by ODA will be vulnerable to modifications.

5.3 Visa Kernel

We next list the flaws discovered in the Visa kernel.

5.3.1 CTQ is Not Protected

In Visa transactions, the **CTQ** tells the terminal the card’s CVM requirements and capabilities. The **CTQ** is authenticated through the signature **SDAD**. However, the **CTQ** is not protected for transactions without fDDA. This flaw was discovered by [29] and exploited in the PIN-bypass attacks by [29, 6, 36, 5], see §6.5.1, §6.5.2, §6.6.1, and §6.7.2.

5.3.2 No Limit for Foreign Currencies

[18] discovered that many Visa credit cards do not require a CVM for high-value transactions in a foreign currency. This allows the *No Cardholder Verification in Foreign Currencies* attack, see §6.2.1.

5.3.3 PIN Verification Over NFC

Encrypted Offline PIN is intended for EMV contact. The terminal encrypts the PIN and sends it to the card, which verifies the PIN. In contrast, in contactless transactions, the PIN is verified by the issuer via Online PIN. [17, 19] showed that the Offline PIN functionality is available over NFC, also offering queries for the remaining PIN attempts. [17, 19] exploit this in their *PIN-Guessing* attacks, see §6.3. Offline PIN was available on Visa cards but not on the tested Mastercard cards.

5.3.4 Merchant Details are Not Authenticated

The card does not authenticate any merchant or terminal data with the MAC **AC**. Thus, the issuer determines the merchant account based on the data from the acquirer.

5.3.5 TTQ is Not Protected

In VISA transactions, the **TTQ** informs the card of the terminal’s CVM capabilities and requirements. Experiments by [29] showed that the **TTQ** could be modified. According to [36], it is unclear whether the **TTQ** is omitted from the **AC** or if the issuer does not check the **TTQ**. This flaw is exploited by the PIN-bypass attacks by [29, 36], see §6.5.1 and §6.7.2.

5.4 Mobile Devices

We next list the flaws in mobile devices such as phones.

5.4.1 Mobile Devices Always Send CDCVM

Google Pay allows for low-value transactions with a locked phone. In contrast, high-value transactions require the phone to be unlocked. In the Visa kernel, the card indicates with the **CTQ** if CDCVM was performed. [29] observed that Google Pay Visa cards always send a **CTQ** indicating that CDCVM was performed if the card accepts the transaction, independently of whether the screen was locked. This is exploited in the PIN-bypass attack on Google Pay, see §6.7.1

5.4.2 Transit Mode Initiated with "Magic Byte"

[36] discovered that transit terminals send some "magic bytes" to unlock Apple Pay and initiate a transit mode transaction. These "magic bytes" can be recorded and replayed. [36] exploits this in their PIN-Bypass attack, see §6.7.2.

6 Attacks on EMV

In this section, we show how the flaws just presented are exploited by surveying the main attacks on EMV contactless reported in the literature. For each attack, we state the violated security properties (P1-P6) and the required adversary capabilities (A1-A8), as presented in §3 and §4, which we also list in Table 1 in Appendix C. As each attack requires controlling the NFC channel (A1), we will not list this capability here.

We structure the attacks according to seven classes: card cloning, replay, PIN-guessing, denial of service, PIN-bypass, merchant-holding-the-bag, and downgrade attacks. We further structure the PIN-bypass attacks into attacks on Visa, Mastercard, and mobile devices. In Appendix §B.3, we describe the attacks on mag-stripe mode.

6.1 Card Cloning

In a card cloning attack, a physical card is created that mimics the target card and is accepted in transactions.

6.1.1 Magnetic Stripe Cloning

The attack by [28] extracts data from EMV mode transactions to create a magnetic stripe card. An adversary records the unencrypted **Track 1** and **Track 2 Equivalent Data**, see §5.1.5, and extracts the magnetic stripe data from it, see §5.1.12. This data is written on a magnetic stripe card, forging a magnetic stripe card for the victim's account. This attack is viable as the magnetic stripe is still supported, see §5.1.13, and as magnetic stripes mostly use the insecure paper signature, see §5.1.7. This attack extracts and misuses data from an EMV contactless transaction, violating the secrecy property (P5). This attack does not violate further properties, as it does not forge an EMV contactless transaction but a magnetic stripe transaction.

6.2 Replay Attacks

In a replay attack, also known as skimming, eavesdropping, or pre-play, transaction data is recorded and reused later.

6.2.1 No Cardholder Verification in Foreign Currencies

The attack by [18] exploits the lack of cardholder verification for foreign currencies, see §5.3.2. The authors exploit this flaw in a replay attack. The adversary records a transaction between a terminal emulator and a target card. Afterwards, the transaction is sent to a rogue merchant (A5) who sends the transaction to the acquirer who forwards it to the issuer. If the issuer accepts the transaction, the funds are transferred from the victim cardholder's bank account to the rogue merchant's account. As the card does not authenticate the merchant details to the issuer, see §5.3.4, the recorded transaction data could be used by any rogue merchant. This attack violates the cardholder's intent (P3) and bypasses the CVM (P3.1).

[18] implemented a proof-of-concept rogue merchant, that was not tested against a live system. Hence, the attack's practicality is unclear. It is unclear too if this flaw could be exploited in a MITM attack or if it would be more practical to directly use a stolen card abroad. [18] argues for a replay attack as no synchronization is required, in contrast to a MITM attack.

6.2.2 Replay with Nonce Reuse

[29] record a transaction between a card and a terminal emulator for a given $\mathbf{UN_T}$. This transaction can then be replayed to a modified terminal (A5) that always uses this $\mathbf{UN_T}$.

The fresh nonce $\mathbf{UN_T}$ from the terminal and the counter **ATC** from the card should prevent replay attacks. However, [29] demonstrated that the issuer does not check if a terminal uses the same $\mathbf{UN_T}$ multiple times and that some issuers accept out-of-order **ATC**s, see §5.1.8 and §5.1.9. This attack violates the cardholder's intent (P3) and the stronger notions of data authentication (P1) required to rule out replay attacks.

6.3 PIN Guessing

In the *PIN-Guessing* attacks by [17, 19], the adversary automatically guesses and tests different PINs on cards. A "guessing device" is placed at an entrance that requires NFC access cards. The attack relies on victims placing their entire wallet at the NFC reader and thus in NFC range of the guessing device. Guessing the PIN correctly violates the secrecy of the PIN (P5). An adversary could then steal the card and use it at a POS terminal, violating the cardholder's intent (P3).

This attack exploits the Offline PIN functionality over NFC, see §5.3.3. The guessing device can check the remaining PIN attempts to avoid blocking the card. If no attempts are left, the attack can be delayed until the cardholder resets the remaining attempts at a POS terminal or ATM. [19] extend this attack by guessing the PINs of different cards in a wallet individually.

6.4 Denial of Service Attacks

A denial of service (DoS) attack prevents the correct usage of the payment system and violates availability (P6).

PIN-Guess-Spamming [17] propose using their *PIN-Guessing* attack as a DoS attack by querying PINs until the attempt limit is reached, blocking the card. The adversary's motivation could be to be a nuisance or blackmail issuing banks by threatening to upset customers [19].

6.5 PIN-Bypass Attacks on Visa

High-value transactions require cardholder verification, which is for physical cards mostly Online PIN. With a PIN-bypass attack however, the adversary can perform a high-value transaction without providing the PIN (P3.1), which violates the cardholder's intent (P3). [17] explain how PIN-bypass attacks could be performed without providing a concrete attack: A MITM attacker can modify control data such as the CVM.

6.5.1 Combined TTQ, CTQ Modification Attack

The first PIN-bypass attack on contactless transactions was described by [29]. The attack modifies the **TTQ** and **CTQ**, which are used to choose the CVM. The adversary first modifies the **TTQ** from the terminal to indicate that no cardholder verification is required. The adversary then modifies the **CTQ** from the card to indicate that CDCVM was performed. As a result, the terminal will assume that it is talking to a phone that performed CDCVM and it will not require PIN verification. Thus, the card and terminal disagree (P1) on the **TTQ** and **CTQ**.

These modifications are possible as both the **TTQ** and **CTQ** are not cryptographically protected in Visa transactions without fDDA, see §5.3.5 and §5.3.1. [29] and [36] point out that this attack could be detected and prevented by checking that CDCVM is not performed by plastic cards, see §5.1.11.

6.5.2 CTQ Modification Attack

The PIN-bypass attack by [6] is similar to the attack by [29]. The attack also targets Visa transactions without fDDA. This attack only requires modifying the **CTQ** and not the **TTQ**. However, an additional bit of the **CTQ** is modified, indicating to the terminal that Online PIN verification is not required.

6.6 PIN-Bypass on Mastercard

The PIN-bypass attacks on Visa just described are not directly possible on Mastercard's kernel 2 as the **AIP**, which indicates the supported and required CVMs, is cryptographically protected through the **AC** MAC and, if present, the **SDAD** signature. In this section, we describe two PIN-bypass attacks that also violate cardholder intent (P3) and bypass the PIN (P3.1).

6.6.1 Card Brand Mixup

[5] trick the terminal into running the Visa kernel with a Mastercard card. This lets the adversary to perform the *CTQ Modification* PIN-bypass attack, see §6.5.2, with a Mastercard card.

The card and terminal choose a kernel according to the **AID**s sent. As the **AID**s are not cryptographically protected, see §5.1.10, the adversary can modify the card-sourced **AID**s to indicate that the card only supports the Visa kernel. The terminal will thus choose the Visa kernel and respond with the Visa **AID**. The adversary replaces it with the Mastercard **AID**, resulting in a disagreement (P1) on the **AID**. The terminal therefore runs the Visa kernel while the card runs the Mastercard kernel. The adversary harvests the terminal's Visa commands to build Mastercard commands for the card and vice versa for the card's responses, see §5.1.2.

Experiments by [5] showed that this attack is possible with some terminals, whereas, other terminals decline modified transactions. The declined transactions were probably detected by the issuer using additional checks on the transaction data. As part of the disclosure process of [5], Mastercard implemented measures on their network to mitigate this attack. The **AID** is checked against the **PAN**, see §5.1.3. Experiments by [5] showed the effectiveness of those measures.

6.6.2 Inducing Authentication Failure

[7] discovered another PIN-bypass attack on the Mastercard kernel that exploits authentication failures. The attack enables the modification of all data that is not authorized online through the **AC** by the issuer. This includes the **CVM List** that informs the terminal of the card's supported CVMs. The PIN can be bypassed by either replacing the **CVM List** with an empty list or replacing Online PIN with Paper Signature. Using a phone as the card emulator enables the attacker to display their own signature on the screen, so the cashier can verify the attacker's signature, see §5.1.7.

This authentication failure is induced by replacing the **CA Public Key Index** with an invalid value. Thus, the terminal fails to access the **CA Public Key**, it does not request the **SDAD**, see §5.2.1, and the **CVM List** is not authenticated. The ‘CDA Failed’ bit in the **TVR** is thus set. If this bit is also set in the **IAC-Denial** or **TAC-Denial**, the terminal declines. However, the adversary can replace the **IAC-Denial** from the card and many terminals have this bit in the **TAC-Denial** set to zero, see §5.1.4. This results in the terminal not declining and not authenticating the **CVM List** and thus accepting the modified CVMs. Thus, the terminal and card disagree (P1) on the **CA Public Key**, **CVM List**, and **IAC-Denial**.

6.7 CVM-Bypass on Mobile Devices

Mobile devices offer CDCVM. In a CVM-bypass attack, an adversary pays with a locked phone and bypasses CDCVM (P3.1), violating cardholder intent (P3).

6.7.1 TTQ, CTQ Modification Attack on Google Pay

The PIN-Bypass attack by [29], see §6.5.1, could also be applied to Google Pay. Low-value Google Pay transactions do not require CDCVM and they can be performed with a locked phone. In contrast to the attack on plastic cards, in the Google Pay attack only the **TTQ** must be modified. In the plastic card attack, the **CTQ** is modified to indicate that CDCVM was performed. This modification is not required when attacking Google Pay as mobile devices always send *CDCVM performed* when they accept a transaction, see §5.4.1.

6.7.2 “Magic byte” PIN-Bypass

[36] exploit Apple Pay’s transit mode and the Visa kernel. Replaying some “magic bytes” unlocks Apple Pay on a locked iPhone, see §5.4.2. This allows relaying a transit mode transaction between a initially locked iPhone and a transit terminal.

To relay a non-transit transaction to a non-transit terminal, [36] modified the unprotected **TTQ**, see §5.3.5, such that the *ODA for Online Authorizations supported* bit and the *EMV mode supported* bit are set. [36] observed a low success rate due to the timings of the relayed messages. The success rate could be improved by caching the READ RECORD response.

The above modifications enable the attacker to relay a low-value transaction between a non-transit terminal and a locked iPhone. The attack can be extended for high-value transactions similarly to the PIN-bypass attack on Visa cards by [6], see §6.5.2. The adversary modifies the unprotected **CTQ**, see §5.3.1, to indicate that CDCVM was performed. It is modified in the GET PROCESSING OPTIONS response as well as in the READ RECORD response. This results in a disagreement (P1) between the card and terminal on the **TTQ** and **CTQ**.

Mastercard transit transactions are protected by an additional check that the Merchant Category Code (**MCC**) indicates a transport service. The **MCC** is authenticated by the **SDAD**. Thus, for Mastercard, this attack is limited to relaying to a transit terminal. The Visa kernel does not send the **MCC** to the card but sends it to the acquirer who forwards it to the payment network.

Apple proposed to prevent the attack on Visa's side by checking the **MCC** for Apple Pay transactions without CDCVM (indicated by the **IAD**, see §2.4),

Experiments showed that Samsung Pay always allows for transit mode transactions. However these are only allowed for a value of zero, which makes the attack useless.

6.8 Merchant-Holding-The-Bag

Offline authorized transactions are sent to the issuer after the terminal has accepted the transaction and the cardholder has likely left the shop. [6] report an attack on such transactions where the attacker can use their card at a terminal that accepts the transaction and walk out with the goods. However, the bank declines the transaction later and the attacker is not charged for the goods. Thus, the merchant is left "holding the bag." This violates the no delayed decline property (P2).

This attack targets low-value transactions with the ODA method SDA and DDA. The MAC **AC** is modified, which is not authenticated by SDA and DDA, see §5.1.6, and the terminal cannot verify the **AC** because it does not have access to the symmetric key **mk**. This leads to a disagreement (P1) between the card and terminal on the **AC**. After the terminal authorized the transaction offline and the attacker walked out with the goods, the bank detects the invalid **AC** and declines the transaction. Note that this attack is not possible for transactions with CDA as CDA authenticates the **AC**.

6.9 Downgrade Attacks

A downgrade attack tricks the protocol participants into executing a vulnerable version of the protocol instead of a more secure version that they both support. This enables an adversary to perform an attack on the vulnerable version.

6.9.1 Maestro to Mastercard

PIN-Bypass attack by [7], see §6.6.2, could not target Maestro cards directly. Maestro cards run the Mastercard kernel but with different configuration data than Mastercard cards. The attack could however be combined with an attack similar to the *Card Brand Mixup* attack by [5], see §6.6.1. The unprotected **AID**s, see §5.1.10, are modified so that the terminal runs the Mastercard flow while the card runs the Maestro flow, leading to a disagreement (P1) on the **AID**.

7 Discussion

We discuss here some of our findings regarding the flaws discovered and the methods used to find attacks.

7.1 Flaw Categories and Causes

We now highlight several observations we made while categorizing these flaws and the resulting attacks.

7.1.1 Authentication

Although the flaws listed in §5 vary considerably, missing data authentication is particularly prominent. Attacks become possible when even just one data field is not authenticated, such as the **CTQ** in §5.3.1, the **TTQ** in §5.3.5, or the **AID** in §5.1.10.

EMV provides two means to authenticate data: the signature-based ODA checked by the terminal and the MAC-based OTA checked by the issuer. EMV inconsistently handles which data is authenticated by which mechanism. For example, for Visa transactions, OTA should protect the **TTQ**, but it does not protect the **CTQ**. In contrast, ODA protects the **CTQ**, but it does not protect the **TTQ**.

In addition to the differences in signature and MAC authentication, the signature based ODA methods themselves differ in the data they authenticate. EMV produces different levels of guarantees when using SDA, DDA, and CDA. Moreover, the most common ODA methods of the different kernels vary, namely CDA for Mastercard and fDDA for Visa, and do not authenticate the same data.

7.1.2 Specification Ambiguities

Multiple flaws stem not from mistakes in the specification but rather they arise from optional configuration options or aspects outside of the public specification, such as the issuer's behavior. For example, incrementing the **ATC** should prevent out-of-order transactions. However, not all issuers check the **ATC**, see §5.1.8. Another example is that some terminals set the 'CDA Failed' bit in the **TAC-Denial** to zero, see §5.1.4, which results in fraudulent transactions being accepted.

7.1.3 Sources of Flaws

Flaws in the EMV protocol can be critical and enable criminals to steal money from banks, cardholders, and merchants. So why have so many flaws in the EMV protocol been discovered? And why has the situation seemingly not improved over time? Here we list potential reasons for this.

First, the requirements on the payment system are not well defined and differ between stakeholders, see §3. A prime example of this is that each payment network has its own kernel and the many configuration options. Moreover, stakeholders must find a trade-off between different requirements, such as keeping the transaction success rate high while preventing malicious transactions, see §3.1. For example, setting the **TAC-Denial**'s 'CDA Failed' bit to one might prevent fraudulent transactions but it might also reject valid transactions.

Second, as with many protocols, EMV has grown organically over time. EMV contactless is based on EMV contact. Supporting backward compatibility for EMV contact within the new context of NFC-based contactless transactions also introduced new problems, such as the availability of offline PIN verification, see §5.3.3. Other problems arise as modern payment cards support EMV contactless and still often support even the vulnerable magnetic stripe, see §5.1.13.

The third and final reason is the high complexity of the EMV contactless protocol. This complexity arises from the protocol's historical development and the numerous kernels and configuration options. This complexity leads to both design and implementation issues and it complicates testing and other measures that could help eliminate flaws.

7.2 Attack Discovery and Prevention

This SoK investigates flaws and attacks on EMV contactless that were discovered and publicized. These flaws and attacks were often published with suggested fixes to prevent the particular attacks. These fixes are mostly lightweight and intended to be implemented with minimal protocol modifications.

7.2.1 Attack Discovery

As observed before, identifying attacks on EMV contactless is challenging given its high complexity. We describe some of the methods that were used to discover the attacks described in §6. We specifically highlight two general approaches to attack discovery: testing and model checking.

Testing attacks on live systems raises ethical questions. The *Merchant-Holding-The-Bag* attack by [6], described in §6.8, was not tested as it would defraud the merchant. In contrast, PIN-bypass attacks [6, 5, 7, 36] were tested using the researchers' own cards to avoid defrauding others. However, [7] point out that, even in these experiments where no one is defrauded, legal questions may arise. They experienced that their accounts at terminal providers were locked and that their cards would be rejected by certain types of terminals.

To avoid ethical and legal issues, some researchers perform experiments on their own terminal implementations [17]. For example, [18] find attacks by testing their own implementation, rather than testing on real cards in the wild. This eliminates potential ethical issues, but leaves open the question of the effectiveness of the attacks in the wild.

In general, testing live systems and testing one's own implementation suffice to discover attacks. However, these methods cannot prove that a property holds for a protocol, meaning that a protocol meets some well-defined specification. More formal approaches can not only identify attacks but also verify protocol properties with respect to a given protocol model. [6] created a comprehensive formal model using the state-of-the-art verification tool Tamarin [38, 34]. Tamarin can, given a protocol model, desired security properties, and an adversary model, provide formal proofs for those properties that hold as well as find attacks on those properties that do not hold. The analysis by [6] revealed insecure configurations and corresponding attacks. Moreover, they also produced machine-checked proofs for alternative secure configurations. Tamarin can also be used to verify proposed fixes [6, 5, 36, 7].

Note that formal method tools such as model checkers only prove properties of an abstract protocol model. The model might not cover all possible configurations and protocol details and hence they might not capture all possible attacks. For example, the models by [6] could not capture the *Card Brand Mixup* attack by [5] as they did not account for transactions performed by the VISA kernel being forwarded to the Mastercard network. Later, the original model was extended to capture such forwarding and to verify the proposed fixes. Similar extensions were made for other attacks [36, 7]. In contrast, some vulnerable configurations might be part of a model but there might not be any real-world card or terminal with this configuration. Thus, it is crucial to validate attacks with experiments on actual systems.

Summing up, the combination of both formal verification tools as well as testing live systems is needed to discover attacks, verify fixes, and prove security properties.

7.2.2 Cat and Mouse Game

To prevent a given attack, the flaws that enable the attack do not necessarily need to be fixed. The attack could also be prevented by introducing new measures. For example, the *Card Brand Mixup* attack [5], see §6.6.1, exploits the unprotected **AID**, see §5.1.10. Mastercard implemented checks on their network to prevent this attack. However, the unprotected **AID** remained and it was subsequently exploited by the *Maestro to Mastercard* downgrade attack [7], see §6.9.1.

We see an unfortunately all too familiar escalating battle between the discovery of new attacks and the incorporation of fixes. Some flaws seem to be exploited repeatedly and fundamental problems such as the lack of authentication and verification of important data seem to remain in the protocol.

7.2.3 Vision

Is there any escape from the seemingly never ending spiral of penetrate and patch? This problem seems to result from, or at least to be magnified by, the tremendous complexity and the need for backwards compatibility of EMV contactless. Reducing complexity by designing a new protocol without backwards compatibility appears unrealistic for a protocol that is as widely deployed and that has as many different stakeholder requirements as EMV does. Thus, EMV contactless's complexity will likely remain.

We believe that the security of such a complex protocol can only be guaranteed using formal methods. At minimum, verification of existing protocols should be carried out. In the ideal world, security critical protocols such as EMV should be built, hand-in-hand, with formal models and proofs of their correctness. Different methods exist to achieve this. Some methods analyze protocol models, such as symbolic verification tools like Tamarin, and ProVerif [9], while others aim to create verified code [40, 8, 1]. Experience building and verifying models of substantial protocols like EMV[6, 5, 4], TLS 1.3 [12], 5G AKA [3] and verification down to code of the TLS 1.3 Record Layer [13] suggest that existing techniques are up to this task.

8 Conclusion

In this SoK, we explored attacks on EMV contactless, categorizing them and identifying the exploited flaws. We also provided a comprehensive framework consisting of the adversary models and security properties, and analyzed attack discovery. Although our focus was on the contactless protocol, it could be extended to other EMV technologies, like contact, payment tokenization, or 3D secure, or to other technologies such as non-EMV chip cards.

Research on modern payment cards will continue. The large number of flaws found in the past suggests that many more flaws and attacks will be found in the future. The new kernel 8 and future technologies require a thorough analysis, some of which is underway [4]. We hope that our framework of adversary models and security properties will be helpful in this regard.

References

[1] Project Everest. `https://project-everest.github.io/`. Accessed: January 2024.

[2] Raja Hasnain Anwar, Syed Rafiul Hussain, and Muhammad Taqi Raza. In Wallet We Trust: Bypassing the Digital Wallets Payment Security for Free Shopping. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 541–558. USENIX Association, 2024.

[3] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A Formal Analysis of 5G Authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1383–1396. ACM, 2018.

[4] David Basin, Xenia Hofmeier, Ralf Sasse, and Jorge Toro-Pozo. Getting Chip Card Payments Right. In *Formal Methods (FM 2024)*, LNCS, pages 29–51. Springer, 2025.

[5] David Basin, Ralf Sasse, and Jorge Toro-Pozo. Card brand mixup attack: Bypassing the PIN in non-Visa cards by using them for visa transactions. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 179–194. USENIX Association, 2021.

[6] David Basin, Ralf Sasse, and Jorge Toro-Pozo. The EMV Standard: Break, Fix, Verify. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1766–1781. IEEE, 2021.

[7] David Basin, Patrick Schaller, and Jorge Toro-Pozo. Inducing Authentication Failures to Bypass Credit Card PINs. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3065–3079. USENIX Association, 2023.

[8] Karthikeyan Bhargavan, Abhishek Bichhawat, Quoc Huy Do, Pedram Hosseyni, Ralf Küsters, Guido Schmitz, and Tim Würtele. DY*: A Modular Symbolic Verification Framework for Executable Cryptographic Protocol Code. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 523–542. IEEE, 2021.

[9] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proceedings. 14th IEEE Computer Security Foundations Workshop, 2001*, pages 82–96. IEEE, 2001.

[10] Yuexi CHEN, Marc Kekicheff, Mustafa Top, and Hao Ngo. Binding cryptogram with protocol characteristics. International Patent App. WO2018136494A1, 2018.

[11] 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.

[12] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A Comprehensive Symbolic Analysis of TLS 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1773–1788. ACM, 2017.

[13] Antoine Delignat-Lavaud, Cedric Fournet, Markulf Kohlweiss, Jonathan Protzenko, Aseem Rastogi, Nikhil Swamy, Santiago Zanella-Beguelin, Karthikeyan Bhargavan, Jianyang Pan, and Jean Karim Zinzindohoue. Implementing and Proving the TLS 1.3 Record Layer. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 463–482, 2017.

[14] Jean-François Dhem, François Koeune, Philippe-Alexandre Leroux, Patrick Mestré, Jean-Jacques Quisquater, and Jean-Louis Willems. A Practical Implementation of the Timing Attack. In *Smart Card Research and Applications*, pages 167–182. Springer, 2000.

[15] Saar Dimer. Chip & PIN terminal playing Tetris | Light Blue Touchpaper. `https://www.lightbluetouchpaper.org/2006/12/24/chip-pin-terminal-playing-tetris/`, December 2006. Accessed: November 2024.

[16] Saar Drimer and Steven J Murdoch. Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks. In *16th USENIX Security Symposium (USENIX Security 07)*. USENIX Association, 2007.

[17] Martin Emms, Budi Arief, Troy Defty, Joseph Hannon, Feng Hao, and Aad Van Moorsel. The Dangers of Verify PIN on Contactless Cards. Technical Report 1332, Newcastle University, 2012.

[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*, CCS '14, 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*, volume 7859 of *LNCS*, pages 313–321. Springer, 2013.

[20] Martin Emms and Aad van Moorsel. Practical attack on contactless payment cards. In *HCI 2011: Health Wealth and Happiness*. Newcastle University, 2011.

[21] EMVCo. EMV® Technologies. `https://www.emvco.com/emv-technologies/`. Accessed: October 2024.

[22] EMVCo. Organisation Structure. `https://www.emvco.com/about-us/organization-structure/`. Accessed: July 2023.

[23] EMVCo. Why EMV®? `https://www.emvco.com/why-emv/`. Accessed: July 2023.

[24] EMVCo. EMV Contactless Specifications for Payment Systems, Books A, B, C-2, C-3, Version 2.1. `https://www.emvco.com/specifications/`, March 2011. Accessed: March 2023.

[25] EMVCo. EMV Contactless Specifications for Payment Systems, Books A, B, C-2, C-3, C-8, E, Version 2.11 (1.1, 1.0). `https://www.emvco.com/specifications/`, June 2023. Acessed: June 2023.

[26] EMVCo. EMV Integrated Circuit Card Specifications for Payment Systems, Books 1-4, Version 4.4. `https://www.emvco.com/specifications/`, October 2022. Accessed: January 2023.

[27] Lishoy Francis, Gerhard Hancke, and Keith Mayes. A Practical Generic Relay Attack on Contactless Transactions by Using NFC Mobile Phones. *International Journal of RFID Security and Cryptography*, 2:92–106, 2013.

[28] Leigh-Anne Galloway. It only takes a minute to clone a credit card, thanks to a 50-year-old problem. `https://drive.google.com/file/d/17YIKh2aWf9n-Hr8cD8_K9vUGtiM-zuFU/view?usp=sharing&ref=leigh-annegalloway.com`, 2020. Accessed: February 2024.

[29] Leigh-Anne Galloway and Tim Yunusov. First Contact: New vulnerabilities in Contactless Payments. In *Black Hat Europe 2019*, 2019.

[30] Vicki Hyman. Swiping left on magnetic stripes. `https://www.mastercard.com/news/perspectives/2021/magnetic-stripe/`. Accessed: December 2022.

[31] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology (CRYPTO '96)*, pages 104–113. Springer, 1996.

[32] Oliver Koemmerling and Markus G Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. *Smartcard*, 99, 1999.

[33] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007.

[34] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification*, volume 8044 of *LNCS*, pages 696–701. Springer, 2013.

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

[36] Andreea-Ina Radu, Tom Chothia, Christopher J.P. Newton, Ioana Boureanu, and Liqun Chen. Practical EMV Relay Protection. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1737–1756. IEEE, 2022.

[37] Michael Roland and Josef Langer. Cloning Credit Cards: A Combined Pre-play and Downgrade Attack on EMV Contactless. In *7th USENIX Workshop on Offensive Technologies (WOOT 13)*. USENIX Association, 2013.

[38] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *2012 IEEE 25th Computer Security Foundations Symposium*, pages 78–94. IEEE, 2012.

[39] Haoqi Shan and Jian Yuan. Man in the NFC. In *Defcon*, volume 25, 2017.

[40] Christoph Sprenger and David Basin. Refining security protocols. *Journal of Computer Security*, 26(1):71–120, 2018.

A Message Sequence Charts

See Figures 9 and 10 for an overview of kernels 2 and 3.

B Mag-Stripe Mode

EMV contactless offers two modes. We previously covered EMV contactless's EMV mode. Here we provide an overview of EMV contactless's mag-stripe mode, its flaws and attacks on it. Note that mag-stripe mode is a mode of EMV contactless and operates over NFC. It is not to be confused with the physical magnetic stripe found on most payment cards.

B.1 Background Mag-Stripe Mode

Mag-stripe mode is based on the data on the magnetic stripe found on most payment cards. During a mag-stripe mode transaction, the card provides the static data that is also stored on the card's magnetic stripe and an additional dynamic verification code over NFC. Mag-stripe mode is a legacy mode. Visa removed mag-stripe mode from their specification in 2016 and it is not part of the new kernel 8. However, the Mastercard specification still contains mag-stripe mode [25].

A mag-stripe mode transaction starts, like EMV mode, with the SELECT and GET PROCESSING OPTIONS command and response. The card indicates with the **AIP** whether it supports EMV mode. If the card and terminal both support EMV mode, it is used. Otherwise, they perform mag-stripe mode.

Similarly to the magnetic stripe, in mag-stripe mode, the card's data is sent as the **Track 1** and **Track 2 Data**. This data contains the PAN, expiry date, service code, and so-called discretionary data. The **Track 1** and **Track 2 Data** is accessed via the READ RECORD command. Note that the EMV mode supports the so-called **Track 2 Equivalent Data**, which is similar to the **Track 2 Data**.

In mag-stripe mode transactions, the card authenticates using the Dynamic Card Verification Code (**CVC3**). However, the **CVC3** does not authenticate the transaction details. To prevent replay attacks, the **CVC3** is derived from the counter **ATC** and the terminal's nonce $\mathbf{UN_T}$.

B.2 Flaws of Mag-Stripe Mode

We next list the flaws discovered in mag-stripe mode.

B.2.1 Weak Random Number Generation

Mag-stripe mode's **CVC3** cryptogram is computed over the terminal-sourced nonce $\mathbf{UN_T}$. [37] discovered a design flaw that reduces the $\mathbf{UN_T}$'s entropy to only 3 decimal digits. The four-byte $\mathbf{UN_T}$ is limited by two factors: the Binary Coded Decimal (BCD) encoding and the size limitation provided by values stored in the card's mag-stripe data file. [37] exploit this weakness in their cloning attacks, see §B.3.2.

B.2.2 AIP Not Protected

The terminal chooses EMV or mag-stripe mode based on the card sourced **AIP**. However, the **AIP** is not protected in mag-stripe mode and can thus be modified by the adversary. The downgrade attack by [37] exploits this, see §B.3.3.

B.2.3 Terminals Retrying Different UNs

[37] discovered that in mag-stripe mode some terminals retry transactions with a different nonce $\mathbf{UN_T}$ if the **CVC3** cryptogram verification fails. They showed that some terminals have this behavior, while others do not. This flaw allows for an optimization of the cloning attack by [37], see §B.3.2.

B.3 Attacks on Mag-Stripe Mode

We next list the attacks on mag-stripe mode. As stated in the initial paragraph of §B, mag-stripe mode is not to be confused with the magnetic stripe.

B.3.1 Eavesdropping on Card Data

The attack by [20] eavesdrops on the transaction between a terminal and a card via a hidden NFC reader placed at a Point-of-Sales terminal. The attacker can record the card number, cardholder name, expiry date, and issue date, as the card sends this data unencrypted, see §5.1.5. This data can be used for card-not-present transactions, such as payments over a website or the phone. This attack violates the secrecy of this data (P5). As it does not forge an EMV contactless transaction, no further properties are violated.

As [20] are not clear on the EMV mode, we presume that this data was retrieved using mag-stripe mode. This presumption is based on the kernel 2 and 3 specifications v2.1 from 2011 [24], which is the earliest version available on the EMVCo website.

Online purchases often require the **CSC**, which is not part of a EMV contactless transaction, but is printed on the card's backside. [17] state that card-not-present transactions can even be performed in many cases without the **CSC**. Otherwise they propose to capture the **CSC** using a hidden camera (A8).

B.3.2 Mag-Stripe Mode Cloning

The *Mag-Stripe Mode Cloning* attack [37] uses skimmed data from a Mastercard mag-stripe mode transaction to create a functional clone supporting mag-stripe mode. The **CVC3** is computed over the card-sourced counter **ATC** and the terminal-sourced nonce $\mathbf{UN_T}$ to prevent replays. However, due to a design flaw, $\mathbf{UN_T}$'s entropy is only 3 decimal digits, see §B.2.1. The experiments of [37] showed that a card could be queried for those 1000 nonces in about one minute. The recorded **CVC3**s can then be stored on the card clone. This attack violates the secrecy (P5) of the mag-stripe mode data and as the card clone can be used in EMV contactless transactions, this also violates the cardholder intent (P3).

The attack can be further improved since some terminals retry transactions with a different $\mathbf{UN_T}$ if the **CVC3** verification fails, see §B.2.3. This allows the adversary to perform the attack without querying the **CVC3** for every possible $\mathbf{UN_T}$. The cloned card can send an error message after receiving a $\mathbf{UN_T}$ that it did not record and it will receive a different $\mathbf{UN_T}$ from the terminal.

As stated above, the **CVC3** cryptogram is also calculated over the counter **ATC**. The authors of [37] assume that issuers record the **ATC**s used for each card to reject out-of-order **ATC**s. This would limit the attacker's window of opportunity. However, [29] showed that the **ATC** is not properly checked by many issuing banks, see §5.1.8.

[37] implemented a proof-of-concept system, consisting of an Android app to record the **CVC3** and a Java Card application for card cloning.

B.3.3 EMV Mode to Mag-Stripe Mode

The *Mag-Stripe Mode Cloning* attack [37], see §B.3.2, relies on the card and terminal choosing mag-stripe mode. However, if the card and terminal both support EMV mode, they would choose EMV mode over mag-stripe mode. The authors extend their attack by tricking the terminal into using mag-stripe mode with a card clone, even though the original card supports EMV mode.

The **AIP** is modified to indicate that no EMV mode is supported. This modification is possible as mag-stripe mode does not authenticate the **AIP**, see §B.2.2. This leads to a disagreement on the **AIP** (P1). In combination with the *Mag-Stripe Mode Cloning* attack, an attacker can use the cloned card at all terminals that support mag-stripe mode, even though the original card supports EMV mode.

C Tables

Table 1 summarizes the attacks presented in this SoK.

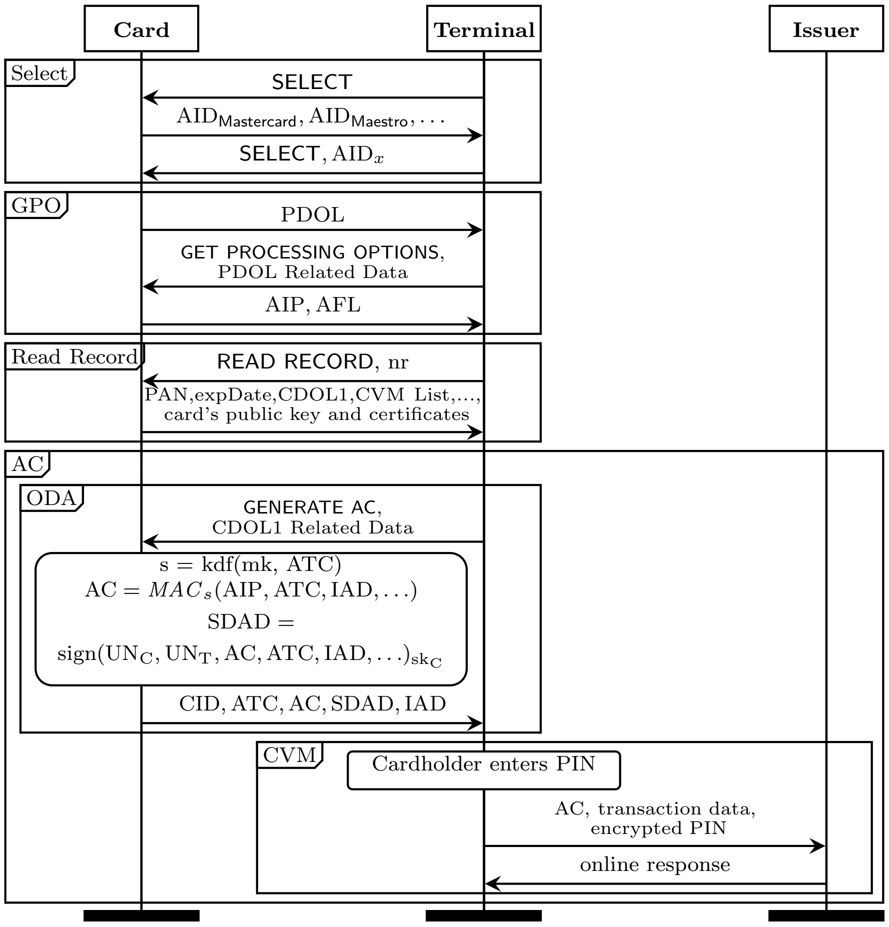
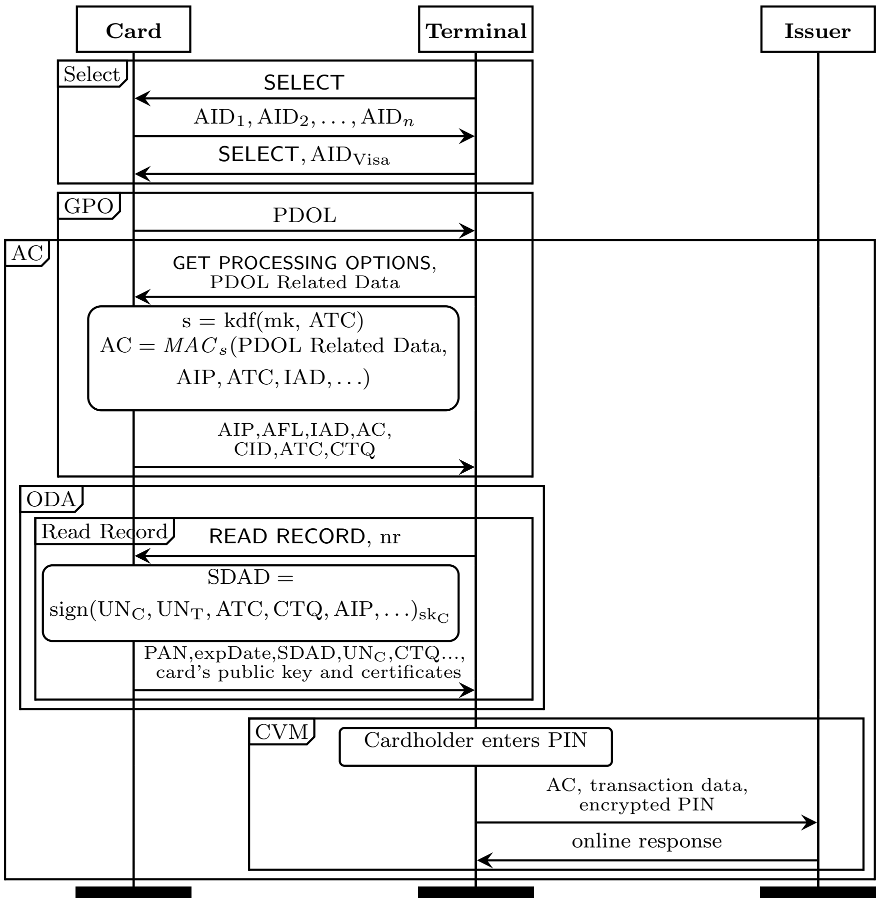

Card
Terminal
Issuer
Select
SELECT
$\text{AID}_{\text{Mastercard}}, \text{AID}_{\text{Maestro}}, \ldots$
SELECT, AID_x
GPO
PDOL
GET PROCESSING OPTIONS, PDOL Related Data
AIP, AFL
Read Record
READ RECORD, nr
PAN,expDate,CDOL1,CVM List,..., card's public key and certificates
AC
ODA
GENERATE AC, CDOL1 Related Data
s = kdf(mk, ATC)
$\text{AC} = MAC_s(\text{AIP}, \text{ATC}, \text{IAD}, \ldots)$
$\text{SDAD} = \text{sign}(\text{UN}_\text{C}, \text{UN}_\text{T}, \text{AC}, \text{ATC}, \text{IAD}, \ldots)_{\text{sk}_\text{C}}$
CID, ATC, AC, SDAD, IAD
CVM
Cardholder enters PIN
AC, transaction data, encrypted PIN
online response

Figure 9: Overview of a Mastercard contactless transaction using the ODA method CDA and the CVM method Online PIN.



Card
Terminal
Issuer
Select
SELECT
$\mathrm{AID}_1, \mathrm{AID}_2, \ldots, \mathrm{AID}_n$
SELECT, $\mathrm{AID}_{\mathrm{Visa}}$
GPO
PDOL
AC
GET PROCESSING OPTIONS, PDOL Related Data
s = kdf(mk, ATC)
AC = MAC_s(PDOL Related Data, AIP, ATC, IAD, ...)
AIP,AFL,IAD,AC, CID,ATC,CTQ
ODA
Read Record
READ RECORD, nr
SDAD = sign($\mathrm{UN_C}, \mathrm{UN_T}$, ATC, CTQ, AIP, ...)$_{\mathrm{sk_C}}$
PAN,expDate,SDAD,$\mathrm{UN_C}$,CTQ..., card's public key and certificates
CVM
Cardholder enters PIN
AC, transaction data, encrypted PIN
online response

Figure 10: Overview of a Visa contactless transaction using its ODA method fDDA and the CVM method Online PIN.

Attack	Violated Security Properties	Adv. Cap.	Flaws	Demon-strated	Patched
6.1.1 Magnetic Stripe Cloning [28]	P5) Secrecy (Track 1, 2 Eqv. Data)	A1) NFC	5.1.13 Mag. Stripes Are Still Supp., 5.1.5 Unencr. Data, 5.1.12 Mag. Stripe Data in EMV, 5.1.7 Weakness of Paper Sign.	in the wild	
6.2.1 No Cardholder Verif. in Foreign Currencies [18]	P3) Cardh. Int., P3.1) High-Value Trans. Require CVM, P3.2) Card Close to Term.	A1) NFC, A5) Term.	5.3.2 No Limit for Foreign Currencies, 5.3.4 Merchant Details Are Not Auth.	in exp. setting	
6.2.2 Replay with Nonce Reuse [29]	P1) Data Auth. (replay) P3) Cardh. Int., P3.2) Card Close to Term.	A1) NFC, A5) Term.	5.1.8 Out-Of-Order ATCs Accepted, 5.1.9 UN Reuse Not Prevented	in the wild	
6.3 PIN Guessing [17, 19]	P5) Secrecy (PIN), P3) Cardh. Int.	A1) NFC	5.3.3 PIN Verification Over NFC	in exp. setting	
6.4 PIN-Guess-Spam. [17, 19]	P6) Availability	A1) NFC	5.3.3 PIN Verification Over NFC	in exp. setting	
6.5.1 Combined TTQ, CTQ Mod. Attack [29]	P1) Data Auth. (TTQ, CTQ), P3) Cardh. Int., P3.1) High-Value Trans. Require CVM, P3.2) Card Close to Term.	A1) NFC	5.3.1 CTQ is Not Prot., 5.3.5 TTQ is Not Prot. 5.1.11 Not Checked if Plastic Cards Do CDCVM 5.1.1 No Relay Prot.	in the wild	No [7]
6.5.2 CTQ Mod. Attack [6]	P1) Data Auth. (CTQ), P3) Cardh. Int., P3.1) High- Value Trans. Require CVM, P3.2) Card Close to Term.	A1) NFC	5.3.1 CTQ is Not Prot., 5.1.11 No Check if Plastic Cards do CDCVM, 5.1.1 No Relay Prot.	in the wild	No [7]
6.6.1 Card Brand Mixup [5]	P1) Data Auth. (AID, CTQ), P3) Cardh. Int., P3.1) High-Value Trans. Require CVM, P3.2) Card Close to Term.	A1) NFC	5.1.2 Visa Resp. Built from Mastercard Resp., 5.1.11 No Check if Plastic Cards Do CDCVM, 5.1.3 No Check if AID and PAN Match, 5.1.10 AID is Not Prot., 5.3.1 CTQ is Not Prot., 5.1.1 No Relay Prot.	in the wild	Yes [5]
6.6.2 Inducing Auth. Failure [7]	P1) Data Auth. (CA PK Index, CVM List, IAC-Denial), P3) Cardh. Int., P3.1) High-Value Trans. Require CVM, P3.2) Card Close to Term.	A1) NFC	5.1.4 TAC-Denial Set to Zero, 5.1.7 Weakness of Paper Sign., 5.2.1 CA Look up Failure Not Declined, 5.1.1 No Relay Prot.	in the wild	No [7]
6.7.1 TTQ, CTQ Mod. Attack on Google Pay [29]	P1) Data Auth.(TTQ), P3) Cardh. Int., P3.1) High-Value Trans. Require CVM, P3.2) Card Close to Term.	A1) NFC	5.3.5 TTQ is Not Prot., 5.4.1 Phones Always Send CD-CVM 5.1.1 No Relay Prot.	in the wild	No [36]
6.7.2 “Magic byte” PIN-Bypass [36]	P1) Data Auth. (TTQ, CTQ), P3) Cardh. Int., P3.1) High-Value Trans. Require CVM, P3.2) Card Close to Term.	A1) NFC	5.3.1 CTQ is Not Prot., 5.3.5 TTQ is Not Prot., 5.4.2 Transit Mode Init. with “Magic Byte” 5.1.1 No Relay Prot.	in the wild	No [36]
6.8 Merchant-Holding-The-Bag [6]	P1) Data Auth. (AC), P2) No Delayed Decline, P3.2) Card Close to Term.	A1) NFC	5.1.6 SDA and DDA do Not Auth. the AC, 5.1.1 No Relay Prot.	No	
6.9.1 Maestro to Mastercard [7]	P1) Data Auth. (AID), P3.2) Card Close to Term.	A1) NFC	5.1.10 AID is Not Prot., 5.1.1 No Relay Prot.	in the wild	No [7]
B.3.1 Eavesdropping on Card Data [20]	P5) Secrecy (data for card not present transactions)	A1) NFC, A8)Visual	5.1.5 Unencr. Data	in the wild	mag-stripe
B.3.2 Mag-Stripe Mode Cloning [37]	P5) Secrecy (mag-stripe mode data), P3) Cardh. Int., P3.2) Card Close to Term.	A1) NFC	5.1.8 Out-Of-Order ATCs Accepted, B.2.1 Weak Random Number Generation, B.2.3 Terminals Retrying Different UNs	in the wild	mag-stripe
B.3.3 EMV Mode to Mag-Stripe Mode[37]	P1) Data Auth. (AIP), P3.2) Card Close to Term.	A1) NFC	B.2.2 AIP Not Prot.	in the wild	mag-stripe

Table 1: Overview of the attacks covered. For each attack, we list the security properties violated, the adversary capabilities required to perform the attack, the flaws that the attack exploits, if the attack was demonstrated in the wild or in an experimental setting and if measures preventing the attack were implemented. The security properties refer to the properties P1) - P6) listed in §3.2. Adversary capabilities A1), A5), A8) refer to §4.1. The column *patched* contains the citation of the latest information available. Note that the attack could have been patched in the meantime. The attacks labeled with *mag-stripe* rely on the legacy mag-stripe mode. The *patched* field is left empty for the attacks for which we do not have any information. We use short-hands such as *Cardh. Int.* for Cardholder Intent, *Auth.* for Authentication or authenticate, *Trans.* for Transaction, *Sign.* for Signature, *prot.* for protected or protection, *Unencr.* for Unencrypted, and *Term.* for Terminal.