Inducing Authentication Failures to Bypass Credit Card PINs

David Basin, Patrick Schaller, and Jorge Toro-Pozo
Department of Computer Science
ETH Zurich

Abstract

For credit card transactions using the EMV standard, the integrity of transaction information is protected cryptographically by the credit card. Integrity checks by the payment terminal use RSA signatures and are part of EMV's offline data authentication mechanism. Online integrity checks by the card issuer use a keyed MAC. One would expect that failures in either mechanism would always result in transaction failure, but this is not the case as offline authentication failures do not always result in declined transactions. Consequently, the integrity of transaction data that is not protected by the keyed MAC (online) cannot be guaranteed.

We show how this missing integrity protection can be exploited to bypass PIN verification for high-value Mastercard transactions. As a proof-of-concept, we have built an Android app that modifies unprotected card-sourced data, including the data relevant for cardholder verification. Using our app, we have tricked real-world terminals into downgrading from PIN verification to either no cardholder verification or (paper) signature verification, for transactions of up to 500 Swiss Francs. Our findings have been disclosed to the vendor with the recommendation to decline any transaction where offline data authentication fails.

1 Introduction

EMV (Europay, Mastercard, VISA) is the standard protocol for modern smartcard payments, with more than 9 billion EMV cards in circulation globally. Due to its significance, it is not surprising that the protocol has been subjected to deep and continued scrutiny, including recent work where the contactless version of the standard was shown to be vulnerable to various critical attacks [5, 6, 19, 24].

The security of EMV transactions relies on both offline and online verification of cryptographic data computed by the payment device (e.g. the credit/debit card, smartphone, or smartwatch) on the transaction data. In the online case, the terminal requests the card issuer's authorization of the transaction through a MAC produced by the card on critical transaction data using a session key, which is derived from a symmetric key shared with the issuer. In the offline case, the terminal validates a card's RSA signature using a PKI, where the root CA certificate is retrieved from the terminal's internal database.

If either of these two authentication mechanisms fails, one would expect the terminal to abort the transaction. However, this is not always the case. For example, the Mastercard specification [15] allows for valid executions where a premature failure of offline authentication does not require the terminal to abort or decline the transaction but instead the terminal ignores all subsequent offline authentication failures during the transaction. This design decision is critical as it allows for data to be modified unnoticed when it is only authenticated offline.

In this paper, we describe a novel attack on Mastercard and Maestro cards that exploits such executions without Offline Data Authentication (ODA) guarantees. In particular, using our attack *one can modify all data that is* ***not*** *included in the MAC generation for online authorization*. This includes data that determines the cardholder verification process, such as the Cardholder Verification Method (CVM) List, which informs the terminal of the CVMs that the card supports.

We explore two modifications of this list that compromise the cardholder verification process:

M1 Suppress all cardholder verification capabilities from the card (i.e. delete the list).

M2 Replace the PIN-based methods with the paper signature method.

Either modification makes the terminal wrongfully believe that the card does not support PIN verification. As a result, a criminal making either modification can use a stolen card to make purchases for large amounts without being asked to enter the card's secret PIN.

Our attack is implemented as a man-in-the-middle (MITM) on top of a relay architecture using two smartphones, as shown

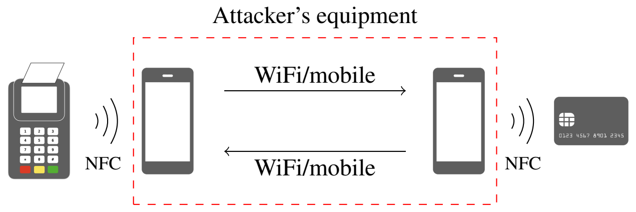


Figure 1: Architecture of a relay attack on contactless payments. From left to right: the payment terminal, card emulator device, Point-Of-Sale (POS) emulator device, and the vicitm's card. The adversary controls the two emulator devices.

in Figure 1. In practice, the attacker holds one of the phones near the victim's card. This could be, for example, because the victim has lost his card, the attacker has stolen it, or the attacker can simply get sufficiently close to the victim. The attacker presents the second phone to the terminal and possibly the cashier. This phone pretends to be running a mobile payment app as depicted in Figure 2. The two phones forward the data exchanged between the victim's card and the terminal and modify it so that the terminal accepts the payment without asking for a PIN, even though the payment amount would require it.

In addition to exploiting authentication weaknesses via a man-in-the-middle, our attack resulting from the modification M2 also exploits user interface weaknesses (UI) associated with the Paper Signature method. In this method, after a successful card-terminal interaction, the cardholder is requested to sign the purchase receipt, which could be either a physical receipt or digitally displayed on a touch screen that can be signed with an electronic pen. The cashier then verifies this against the signature on the payment device, which can either be (a) a physical card, in which case the signature is on its backside, or, if applicable, (b) an electronic device, in which case the signature is displayed on its screen as shown in Figure 2. This distinction is highly relevant for our attack, as explained next.

Whereas a criminal in possession of a lost or stolen card cannot forge the (handwritten) signature on it, for a smartphone-based payment they can forge the signature by simply displaying a made-up payment screen with a cardholder name and signature of the criminal's choice. Clearly this will pass the cashier's verification, regardless of the signature method used (pen&paper versus digital pen&pad) and whether the cardholder's ID is required or not. Our attack thus not only exploits improperly designed failure modes in the EMV protocol, but also user interface weaknesses in digital payment methods.

In summary, our attack resulting from the modification M2 demonstrates how exploits can arise when mechanisms designed in one context, such as a physical signature on the back of a plastic card, are transported into another context, such as an image of a signature lacking a physical context,

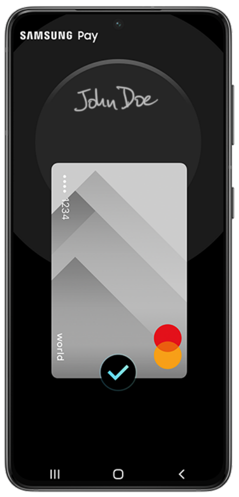


Figure 2: The Samsung Pay screen displaying the cardholder's signature.

that is much simpler to exploit.

Contributions This paper makes the following contributions. First, we identify a novel attack that renders the PIN useless for Mastercard-branded cards. Concretely, our attack deceives the terminal into believing that the payment source does not support any PIN-based Cardholder Verification Method (CVM). As a result, the security offered by these CVMs is not guaranteed, even for high-value transactions.

Second, we demonstrate that this attack is feasible in practice by developing and successfully using live an Android app that implements our man-in-the-middle attack. We have conducted numerous PIN-less transactions for amounts above the CVM-required limit on multiple real-world payment terminals with different cards. We also present countermeasures that can be deployed on the different components involved in the transaction and do not alter the existing execution flow for Mastercard transactions.

Organization We provide technical background on the EMV contactless payment protocol in Section 2, focusing on the Mastercard version. In Section 3, we describe our novel attack in detail, report on our proof-of-concept implementation and experiments, and describe countermeasures. We also show how our attack can be found using an slight modification of our previous formal model of the EMV protocol and we formally prove the security of our countermeasures. Afterwards, in Section 4, we elaborate on previously discovered weaknesses in the EMV standard and make comparisons. Finally, we draw conclusions in Section 5.

Ethics and Disclosure We initiated a responsible disclosure process with Mastercard on January 10th, 2022. In a follow-up meeting with them, we described our findings in detail and recommended countermeasures to prevent our attack. As requested by Mastercard, we have also shared this draft with them. In all the tests we performed, we only used our own cards. No cardholders, merchants, or issuers have been defrauded.

2 Background

The EMV ecosystem involves multiple stakeholders. At its core are payment networks (e.g. Mastercard and VISA), which provide the infrastructure for payment transactions from cardholders' accounts to merchants' accounts. A financial institute (typically a bank) can become part of this ecosystem by issuing cards that comply with the EMV standards. Merchants, selling goods and services, can run payment terminals that implement the EMV standard and that are connected to the payment networks.

In order to secure transactions and ensure that only legitimate ones are accepted by the networks, various security measures are in place. In this work, we focus on the security properties of the EMV communication protocols and its elements. To secure communication between the card, the payment network, and the card-issuing bank, there is a global, hierarchical Public Key Infrastructure (based on RSA keys) in place. A set of root Certificate Authorities accepted by the payment networks provide certificates for card issuers who in turn provide certificates stored on the cards they issue. By possessing the root Certificate Authorities' public keys, terminals can verify the correctness of the cards' certificates and authenticate them as legitimate cards. In addition to asymmetric keys, which are mainly used for offline verification, the cards hold symmetric keys that are shared only with the issuing bank. Last but not least, the legitimate use of each card is ensured by cardholder verification using a PIN, a handwritten signature, or a smartphone.

Another security element of card payment systems is fraud detection. This may, for example, involve the behavioral analysis of customers and anomaly detection. Such fraud detection mechanisms build an additional layer of defence and can invalidate technically legitimate transactions. These types of security measures are, however, not in the focus of this work.

The EMV protocol for contactless transactions extends the card-reader based protocols and is specified in over 1,200 pages of documentation. In this section we summarize this protocol, focusing on those elements relevant to our attack. A glossary of EMV acronyms is given at the end of this article.

2.1 The Protocol

EMV supports six provider-specific contactless protocols that define the data elements employed and how data is exchanged between the issuer, the terminal, and the Integrated Circuit Card (ICC, commonly known as *the card*). In addition to the six different payment networks that must be supported, there are issuer-specific data elements that are required, depending on the ICC's issuer.

The Integrated Circuit Card and the terminal communicate using a sequence of command-response pairs. Commands are sent by the terminal, processed on the ICC, and they result in a specific response by the ICC. Commands and responses are sent as Application Protocol Data Units (APDUs).

Data on the ICC is stored in information items that are called files and are ordered in a tree structure on the ICC. The card issuer is responsible for ensuring that the data on the ICC has the correct format. Files can be accessed by the terminal using a `SELECT` command in combination with the corresponding file name. Records in a file are read using the `READ RECORD` command.

From an abstract point of view, a transaction involves the following steps:

1. Application Selection: The terminal and the ICC agree on an EMV contactless protocol that is implemented as a so-called kernel on the terminal.

2. Synchronisation: The terminal and the ICC negotiate options, e.g. the Cardholder Verification Method (CVM) to be used, and they exchange information required to successfully complete the transaction, such as the data elements requested by the card issuer.

3. Cardholder Verification: The terminal verifies that the person making the purchase is the owner of the card.

4. Authentication and Authorization: The terminal and card take actions to decide if the transaction should be declined at this point, or whether further verification methods are required, such as (i) offline data authentication or (ii) online authorization.

In the following sections, we describe in more details those transaction processing steps that are relevant for our attack.

2.1.1 Application Selection

A terminal starts a transaction by sending a `SELECT` 2PAY.SYS.DDF01 command for the Dictionary Definition File (DDF), an element in the tree-structured hierarchy of files on the ICC. The response of a legitimate ICC contains the list of supported applications, identified by their AIDs (Application Identifiers).

On the terminal side, the received AID (or list of AIDs) results in what is called the *Kernel Activation*. This step could also be understood as selecting the protocol to be used for the current transaction between the ICC and the terminal.

The EMV contactless specification for payment systems, Book A [14], lists the following kernels:

- Kernel 2 for Mastercard AIDs,
- Kernel 3 for VISA AIDs,
- Kernel 4 for American Express AIDs,
- Kernel 5 for JCB AIDs,
- Kernel 6 for Discover AIDs, and
- Kernel 7 for UnionPay AIDs.

In this paper, we focus on the Mastercard kernel (Kernel 2, specified in [15]) and depict the communication between the terminal and the card as a message sequence diagram in Figure 3. For an extension of our attack to Maestro debit cards, note that, on the terminal side, Maestro cards are processed with the same kernel, using different initialization of parameters.

2.1.2 Synchronisation between Terminal and ICC

As a result of application selection, the terminal knows where to find the information on the ICC necessary to successfully complete the transaction. In the following messages, the terminal learns the data elements required by the issuer and provides required information to the card. Furthermore, the ICC provides information about the admissible Cardholder Verification Method and provides public keys and certificates used in a later step by the terminal to authenticate the card in a process called Offline Data Authentication (ODA).

As a first step after application selection, the terminal issues a SELECT command to select the corresponding Application Definition File (ADF) on the ICC. The card responds with a File Control Information (FCI) message that includes, for example, the card's language preference and the Processing Data Object List (PDOL), a list of terminal-resident data objects required by the ICC for processing the subsequent GET PROCESSING OPTIONS command. The AID identifies the application on the terminal side, whereas the ADF denotes the corresponding filename for the application on the ICC.

The terminal starts the transaction processing by sending the aforementioned GET PROCESSING OPTIONS command, which includes the data elements requested by the card in the PDOL. The ICC's response includes the Application Interchange Profile (AIP). This specifies the card's available functionality and the Application File Locator (AFL), which identifies the files and records on the card to be used for the transaction.

Given the ICC's capabilities, the terminal accesses the required information on the ICC by using READ RECORD commands for the data elements on the card identified by a corresponding Short File Identifier (SFI) and a record number. There is some flexibility regarding the sequence of these records in the ICCs.

The records read by the terminal include the following data elements (note that for some records we implicitly assume the availability of the corresponding functionality on the ICC):

- Primary Account Number (PAN).
- Card Risk Management Data Object Lists (CDOL1 and CDOL2): Two lists of transaction-related data objects that the terminal will send in a later step for the ICC to sign (offline and online verification).
- The Cardholder Verification Method (CVM) List.
- PKI-related information: the card's and the issuer's PK certificate, and the index of the issuing Certificate Authority (CA).

At this point, the terminal and the ICC have bootstrapped the necessary parameter configuration and fixed their structure as required by the issuer. Also, the terminal has received the certificates and keys to authenticate the ICC offline. Because the hash of the security critical records is contained in the ICC's certificate, the terminal can verify the integrity of these records.

For contactless EMV payments, we have not observed the authentication of static data by the terminal at this stage of the protocol. However, the integrity of static data is verified later in the more general procedure of Offline Dynamic Data Authentication, which also covers dynamic data related to the transaction, such as the payment amount. Details of the Offline Data Authentication (ODA) process are provided later in Section 2.1.4.

2.1.3 Cardholder Verification

The purpose of a Cardholder Verification Method (CVM) is to ensure that the person presenting the ICC is indeed the person to whom the card was issued. The Mastercard specification for contactless payments [15] requires that the ICC provides a CVM List. This list defines the CVMs and the conditions under which the CVM should be applied as defined by the issuer. The CVMs are:

Online PIN: The terminal sends the encrypted PIN that has been entered on the terminal's pad for verification to the issuer.

On Device CVM: Mobile payment apps such as Google Pay, Samsung Pay, or Apple Pay provide the possibility of authenticating the cardholder on the device. Typically, they use a fingerprint reader or face recognition for this purpose.

Paper Signature: The cardholder signs the purchase receipt (digital or printed on paper) and the cashier checks the signature against the signature on the back of the card. Modern payment terminals often provide touch screens

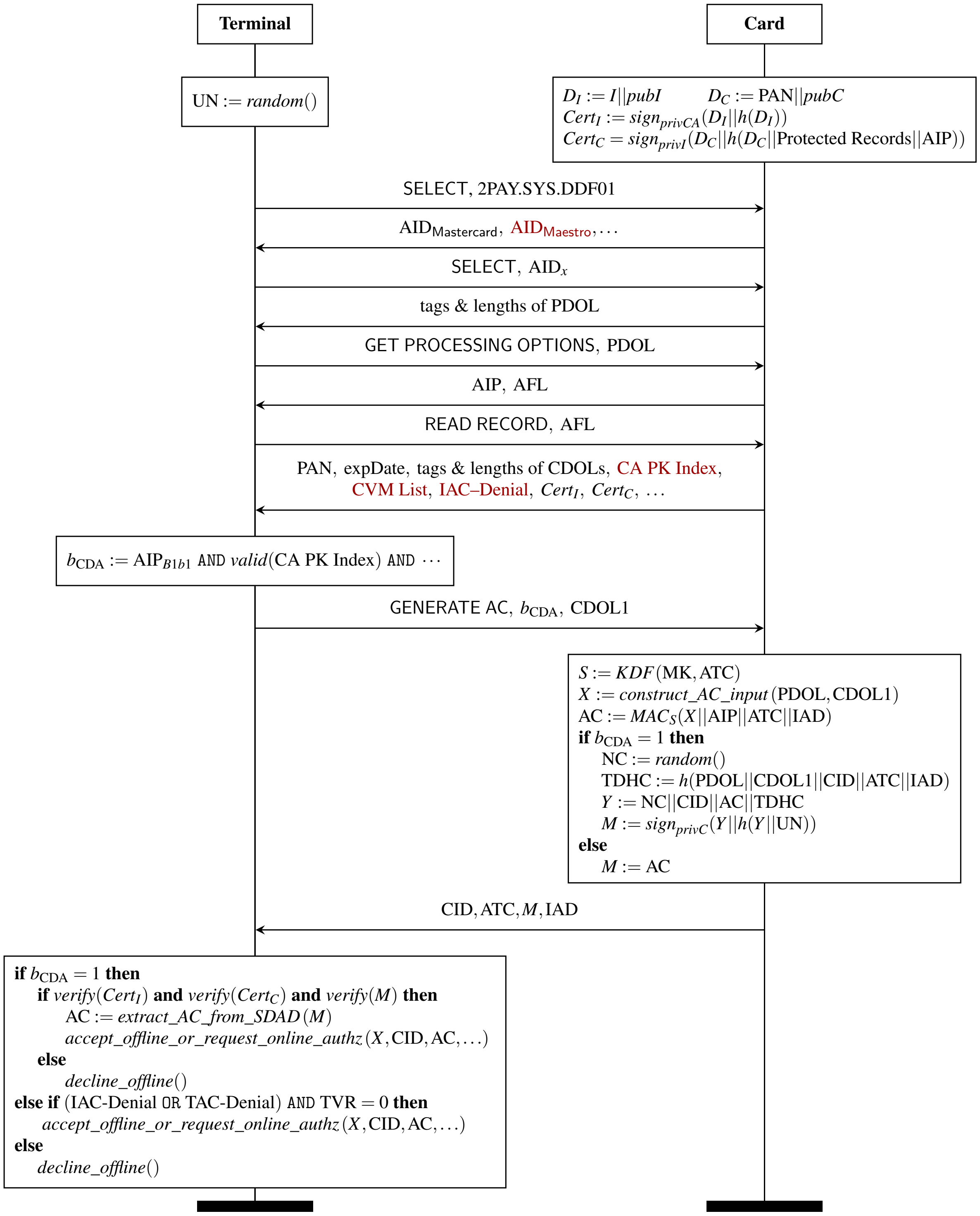

Terminal
Card
UN := random()
$D_I := I||pubI$ $D_C :=$ PAN$||pubC$
$Cert_I := sign_{privCA}(D_I||h(D_I))$
$Cert_C = sign_{privI}(D_C||h(D_C||$Protected Records$||AIP))$
SELECT, 2PAY.SYS.DDF01
$\text{AID}_{\text{Mastercard}}$, $\text{AID}_{\text{Maestro}}, \ldots$
SELECT, AID_x
tags & lengths of PDOL
GET PROCESSING OPTIONS, PDOL
AIP, AFL
READ RECORD, AFL
PAN, expDate, tags & lengths of CDOLs, CA PK Index,
CVM List, IAC–Denial, $Cert_I$, $Cert_C$, …
$b_{\text{CDA}} := \text{AIP}_{B1b1}$ AND $valid$(CA PK Index) AND $\cdots$
GENERATE AC, b_{CDA}, CDOL1
$S := KDF(\text{MK}, \text{ATC})$
$X := construct\_AC\_input(\text{PDOL}, \text{CDOL1})$
$\text{AC} := MAC_S(X||\text{AIP}||\text{ATC}||\text{IAD})$
if $b_{\text{CDA}} = 1$ **then**
$\text{NC} := random()$
$\text{TDHC} := h(\text{PDOL}||\text{CDOL1}||\text{CID}||\text{ATC}||\text{IAD})$
$Y := \text{NC}||\text{CID}||\text{AC}||\text{TDHC}$
$M := sign_{privC}(Y||h(Y||\text{UN}))$
else
$M := \text{AC}$
CID, ATC, M, IAD
if $b_{\text{CDA}} = 1$ **then**
if $verify(Cert_I)$ **and** $verify(Cert_C)$ **and** $verify(M)$ **then**
$\text{AC} := extract\_AC\_from\_SDAD(M)$
$accept\_offline\_or\_request\_online\_authz(X, \text{CID}, \text{AC}, \ldots)$
else
$decline\_offline()$
else if (IAC-Denial OR TAC-Denial) AND TVR $= 0$ **then**
$accept\_offline\_or\_request\_online\_authz(X, \text{CID}, \text{AC}, \ldots)$
else
$decline\_offline()$

Figure 3: Overview of the card-terminal interaction during a Mastercard contactless transaction, where the PDOL and the CDOL1 typically include transaction data, such as the Unpredictable Number (UN), the amount, and the Terminal Verification Results (TVR). **Notation:** $||$ is the concatenation operator; h is the SHA-1 hash function; KDF is a key derivation function; $(privC, pubC)$, $(privI, pubI)$, and $(privCA, pubCA)$ are the private/public key pairs of the card, the issuer, and the Certificate Authority, respectively; b_{CDA} is a flag that indicates whether the terminal requests Combined Dynamic Data Authentication (CDA); $sign_k(m)$ is the signature on m with the key k; $MAC_k(m)$ is a keyed MAC on m with the key k; and AND and OR are the corresponding bitwise operators. Here we have highlighted in red the data objects that the attacker modifies.

for cardholders to sign transactions using their finger or digital pens.

If the transaction amount exceeds a predefined threshold (called the CVM-required limit), the terminal will request verification of the cardholder. The method(s) chosen by the terminal depends on the CVM List, which is supplied by the ICC in one of the records read from the application file.

Clearly, the different CVMs impose different levels of difficulty for the illegal use of a card. It is much easier to convince the cashier of the legitimacy of a forged signature than to guess a PIN correctly, especially when the card is locked after three failed attempts. Things get even worse when the signature is entered on a terminal's touch screen, where it often seems impossible to determine the signature's authenticity.

In this paper, we show that it is possible to modify the CVM List provided by the card in a way that the terminal requests either no CVM at all, or a signature instead of a PIN. Even worse, the modification of the CVM List does not seem to be detected by the issuer in the transaction authorization phase and thus results in a valid transaction.

2.1.4 Authentication and Authorization

There are two central security mechanisms used to prevent misuse of an ICC: Offline Data Authentication (ODA) and Online Transaction Authorization (OTA). Irregularities in either of these mechanisms can lead to a transaction being declined.

Whereas Offline Data Authentication only involves the terminal and the card, Online Transaction Authorization also involves the card issuer. Both mechanisms implement integrity protection for critical data objects (static and transaction specific). While these mechanisms overlap in that they both protect the integrity of some data, there are other data objects that are only protected by the **offline** mechanism. This is crucial for our attack.

Offline dynamic data authentication (i.e. a type of ODA) is performed by the terminal using digital signatures to identify the ICC and to prove the integrity of critical data elements exchanged between the terminal and the ICC. This step requires the Public Key Infrastructure (PKI) that issues certificates for the public keys of the involved entities.

For EMV, there is a list of Certificate Authorities (CAs) accepted by the terminals that issue certificates for card issuers, who in turn issue certificates for each ICC. The card issuer's certificate, as well as the ICC's certificate, have been read as data elements from the card earlier (see Section 2.1.2). They are verified using the CA Public Key Index, a data element provided by the ICC, which points the terminal to the CA in its list of legitimate CAs.

The EMV standard specifies two forms of offline dynamic data authentication: Dynamic Data Authentication (DDA) and Combined Dynamic Data Authentication (CDA). The ICC announces, in the AIP, its supported forms of offline authentication. As it is the most common method currently in use, we focus on CDA here, where the ICC proceeds as follows:

- Create a random nonce NC.
- Concatenate the data elements PDOL, CDOL1, Cryptogram Information Data (CID), Application Transaction Counter (ATC), and Issuer Application Data (IAD) and hash them (using SHA-1) to build the Transaction Data Hash Code:

$$\text{TDHC} = h(\text{PDOL}||\text{CDOL1}||\text{CID}||\text{ATC}||\text{IAD}).$$

- The TDHC is then further combined with the Application Cryptogram (AC, described later), the CID, and the nonce NC to build $Y = \text{AC}||\text{CID}||\text{NC}||\text{TDHC}$.
- Finally, Y is concatenated with the hash of the concatenation of Y and the terminal's Unpredictable Number (UN), and then signed with the ICC's private key to produce the Signed Dynamic Authentication Data:

$$\text{SDAD} = sign_{privC}(Y||h(Y||\text{UN})).$$

Note that this signature protects the integrity of all relevant transaction specific data, as well as some static data elements. In addition to the integrity of the data elements protected by this signature, the ICC's certificate not only contains the PAN the card is issued to and the card's public key, but also static data elements of the ICC, such as the CVM List.

In contrast to offline data authentication, which involves the terminal and the ICC, online authorization involves the issuer and the ICC. The security of online authorization relies on a secret key MK, shared between the issuer and the ICC.

For each transaction, the ICC and the issuer derive a session key S based on the Application Transaction Counter (ATC) and MK. The ICC then proceeds as follows:

- Extract the transaction relevant data elements from PDOL and CDOL1 and concatenate the resulting elements together as X.
- Compute the Application Cryptogram

$$\text{AC} = MAC_S(X||\text{AIP}||\text{ATC}||\text{IAD}),$$

where MAC is a block-cipher-based CBC-MAC using the key S.

Note that the signed message in the offline dynamic data authentication case contains the application cryptogram for the online authorization. However, based on the terminal's decision to perform offline authentication or online authorization only, the terminal sends a GENERATE AC command in which the reference control parameter tells the ICC if CDA is required or online authorization only.

3 The Attack and Countermeasures

In this section we describe how to attack contactless cards running the Mastercard protocol by inducing authentication failures. We first present the threat model considered and then describe our attack. Afterwards we report on our proof-of-concept implementation and practical experiments that we conducted with different cards and readers. Finally, we briefly discuss countermeasures.

3.1 Threat Model

For our attack, we consider the following threat model:

1. The attacker is within the NFC range of the victim's contactless card.
2. The attacker is an active attacker on the NFC channel. In particular, the attacker can read, block, and inject messages on this channel.
3. The channel between the payment terminal and the card issuer is secure in that it provides authenticity and confidentiality.
4. The attacker can reproduce the images associated with the graphical user interface of a digital payment app showing the attacker's own (handwritten) signature and name, if necessary.

This threat model is realistic. The attacker may access a victim's card if, for example, it is lost or stolen. In addition, as shown in previous work [5, 6, 24] and later in Section 3.3, it is possible to carry out active man-in-the-middle attacks on the NFC channel using regular smartphones.

The fourth capability of the considered attacker results, in practice, from user interface weaknesses in digital payment apps. Namely, anyone can easily reproduce images of the UI associated with these apps with any signature and name on it, which makes Paper Signature an insecure Cardholder Verification Method (CVM). Note that this CVM was originally intended for plastic cards, where the cardholder's name and signature cannot be so easily forged.

3.2 Exploiting Authentication Failures

As explained in Section 2, during an EMV transaction the payment terminal performs the validation of the card and transaction data using the Offline Data Authentication (ODA) mechanism. This mechanism employs a PKI where the root CA certificate is looked up from the terminal's database. The entry in this database is determined from the following data supplied by the card in the responses to the READ RECORD commands:

- the Registered Application Provider Identifier, which is derived from the Application Identifier (AID), and
- the CA Public Key Index.

The Mastercard kernel specification [15] (p. 255) states that if the supplied CA Public Key Index is not present in the terminal's CA Public Key database, then the terminal shall set the 'CDA Failed' bit of the Terminal Verification Results (TVR). Recall that the TVR is a data object maintained by the terminal throughout the transaction that holds information on the outcome of verification processes, including Offline Data Authentication (ODA). Note that this TVR update occurrs **before** the terminal issues the GENERATE AC command. This is critical, and is indeed the fundamental design flaw that our attack exploits together with user interface weaknesses in digital payments.

According to the kernel specification (p. 435), if the TVR indicates that CDA has failed and the AIP indicates that the payment source does not support On Device CVM, then the terminal shall **not** request the Signed Dynamic Authentication Data (SDAD) in the GENERATE AC command. This means that no ODA is to be performed and therefore all transaction data whose integrity is only protected by the ODA mechanism is vulnerable to adversarial modification.

Our man-in-the-middle attack induces this premature ODA failure mode to modify such unprotected data. It thereby exploits the fact that in this mode the terminal will ignore all (offline) cryptographic verification failures during the transaction. More technically, the attack is composed of the following three steps:

S1 Modify the card's response to the first SELECT command by replacing the Application Identifier (AID, object with tag 4F or 84) with the Mastercard AID A0000000041010.

S2 Modify the terminal's second SELECT command payload by replacing the Mastercard AID with the card's legitimate AID.

S3 Modify the card's responses to the READ RECORD commands as follows:

(a) Replace the CA Public Key Index (object with tag 8F) with an invalid one[1], in our case we use DD.

(b) Delete the CVM List (object with tag 8E) or modify it by replacing all PIN-based CVMs with the Paper Signature method.

(c) Clear the Issuer Action Code (IAC)-Denial (object with tag 9F0E) by replacing it with all zeroes.

The steps S1 and S2 are relevant only for Maestro cards and are intended to deceive the terminal into executing the default Mastercard transaction flow. These two steps override alternative flows, potentially available for Maestro cards,

[1]CA Public Key data is available at https://www.eftlab.com/knowledge-base/243-ca-public-keys.

which are proprietary and thus unavailable to us. The lack of integrity protection for the AID, which makes the Steps S1 and S2 possible, was first reported in our work [5].

Step S3(b) removes all PIN-based CVMs from the available choices for the terminal to perform the cardholder verification. Note that this step can make either modification M1 or M2, as discussed in Section 1. Finally, the effect of Step S3(c) is to replace the value computed by the formula:

$$(\text{IAC-Denial } \texttt{OR} \text{ TAC-Denial}) \texttt{ AND } \text{TVR}, \qquad (1)$$

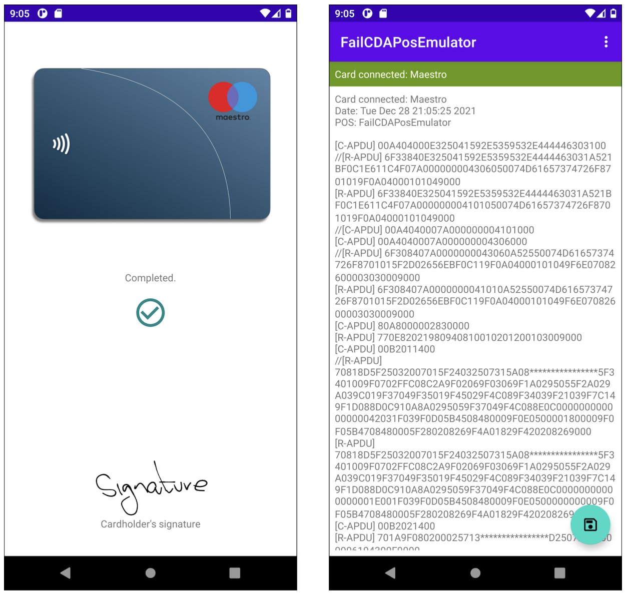
with the value zero, i.e., having zeros in all bit positions, otherwise the transaction is declined offline (recall this from Figure 3). This step is obviously only necessary when the card does not have the IAC-Denial object already cleared. Note that if the terminal has the 'CDA Failed' bit of the TAC-Denial set at the time of reception of the GENERATE AC response, the Terminal Action Analysis would decline the transaction offline (see [15], p. 460) since the formula in (1) would not equal to zero.

3.3 Carrying Out the Attack in Practice

We developed a proof-of-concept Android application to demonstrate the real-world exploitation of the flaws we identified in the Mastercard kernel [15]. Our app implements a man-in-the-middle using two Android phones as previously shown in Figure 1. The two phones communicate with each other using a TCP/IP-based relay channel over WiFi. Our app supports two operation modes: Point-Of-Sale (POS) emulator and card emulator. The POS emulator mode is responsible for the actual modification of messages. The card emulator mode runs our custom host-based card emulation service for Android [3]. Screenshots of our app are displayed in Figure 4.

Our proof-of-concept app requires minimal setup. Namely the relay channel must be configured and the Near Field Communication (NFC) channel between the POS emulator device and the (victim's) credit/debit card must be activated. The attack then works as follows. Once the card emulator is within the NFC range of the payment terminal, it captures the terminal's command, relays it to the POS emulator device, which then modifies it as appropriate and sends it to the card through the previously activated NFC channel. Once the card replies to the command, the POS emulator modifies the response as appropriate, relays it to the card emulator, which then delivers the (possibly modified) response to the payment terminal. This process is repeated for each of the terminal's commands.

Table 1 summarizes the transactions that we have performed using our app. These are all real-world, accepted transactions where Offline Data Authentication (ODA) failed and each of the cards used has Online PIN as the preferred CVM. We successfully circumvented the PIN verification in 9 transactions, using 5 different cards issued by 2 banks from 2 countries. A demo video of the attack being performed for one of these transactions (500 CHF) is available at [1].



```
Card connected: Maestro
Date: Tue Dec 28 21:05:25 2021
POS: FailCDAPosEmulator

[C-APDU] 00A404000E325041592E5359532E444446303100
//[R-APDU] 6F33840E325041592E5359532E4444463031A521
BF0C1E611C4F07A0000000043060500074D61657374726F87
01019F0A04000101049000
[R-APDU] 6F33840E325041592E5359532E4444463031A521B
F0C1E611C4F07A0000000041010500074D61657374726F870
1019F0A04000101049000
//[C-APDU] 00A4040007A000000004101000
[C-APDU] 00A4040007A000000004306000
//[R-APDU] 6F308407A0000000043060A52550074D61657374
726F8701015F2D02656EBF0C119F0A04000101049F6E07082
600003030009000
[R-APDU] 6F308407A0000000041010A52550074D616573747
26F8701015F2D02656EBF0C119F0A04000101049F6E070826
00003030009000
[C-APDU] 80A8000002830000
[R-APDU] 770E82021980940810010201200103009000
[C-APDU] 00B2011400
//[R-APDU]
70818D5F25032007015F24032507315A08***************5F3
401009F0702FFC08C2A9F02069F03069F1A0295055F2A029
A039C019F37049F35019F45029F4C089F34039F21039F7C14
9F1D088D0C910A8A0295059F37049F4C088E0C0000000000
0000042031F039F0D05B4508480009F0E0500001800009F0
F05B4708480005F280208269F4A01829F420208269000
[R-APDU]
70818D5F25032007015F24032507315A08***************5F3
401009F0702FFC08C2A9F02069F03069F1A0295055F2A029
A039C019F37049F35019F45029F4C089F34039F21039F7C14
9F1D088D0C910A8A0295059F37049F4C088E0C0000000000
0000001E001F039F0D05B4508480009F0E0500000000009F0
F05B4708480005F280208269F4A01829F42020826
[C-APDU] 00B2021400
[R-APDU] 701A9F080200025713***************D2507
```

(a) Card emulator (b) POS emulator

Figure 4: Screenshots of our app. The log (partially) displayed in the POS emulator screen corresponds to a transaction of 500 Swiss Francs (CHF) where we downgraded the CVM from Online PIN to Paper Signature. Note that the card emulator screen could have been that of Figure 2.

We also observe that the Mastercard kernel execution flow is very similar to that of the American Express and JCB kernels. This means that our attack should also work for cards of these brands. We did not, however, test this hypothesis due to our limited access to these cards. In the case of VISA cards, we tested our attack (with some minor adaptations specific to this kernel) and the transaction was successful. However, it is unclear whether the induced ODA failures were ignored due to the modified CA PK Index or the terminal simply ignored the card's unverified Signed Dynamic Authentication Data (SDAD). This ignore decision by the terminal has been reported before, e.g. in [6, 24]. We did not test with the rest of the EMV cards (i.e. Discover and UnionPay), as we did not have access to any of them.

3.4 Discussion

The root cause of the vulnerability we describe is that the protocol specification allows for the successful completion of contactless transactions even when the terminal cannot validate the card's Public Key certificate. The protocol's vulnerable execution flow requires pointing the terminal to an invalid index in the root certificate list and the TAC-Denial, a bit-vector internal to the terminal, being zero (see Section 3.2 for details).

The EMV specification [13] (p. 115) recommends *not* setting the TAC-Denial to zero. However, even in cases where

No.	Card	Merchant	Amount (CHF)	PIN bypassed	CVM downgraded to
1	Mastercard Debit	A	**83.95**	Yes	Paper Signature
2		B	**125.00**		
3	Mastercard Debit	A	**133.75**	Yes	No CVM
4	Mastercard Credit	C	1.60	N/A	N/A
5			2.10		
6	Maestro	A	75.35	N/A	N/A
7	Maestro	D	**100**	Yes	Paper Signature
8			**500**		
9	Maestro	E	120	No	N/A
10	Mastercard Debit	E	**120**	Yes	No CVM
11	Maestro	F	**150**	Yes	No CVM
12		G	**100**		Paper Signature
13		G	**100**		No CVM

Table 1: Thirteen transactions performed using our app. All of these transactions were accepted and subsequently debited from the cardholder's account. The eight cards listed here are all different. Here we have highlighted in **bold** the amounts above the local CVM-required limit (80 CHF) for which we bypassed the PIN. N/A stands for not applicable (e.g. the PIN is not required for amounts below the CVM-required limit and thus no bypass applies).

our attack failed, the terminal implementations seem not to follow this recommendation. Settings following the recommendations would have exhibited a different protocol behavior by the terminal, even for unsuccessful attempts. As a consequence, *every terminal implemented according to the specification and not following the recommendation is vulnerable to our attack*. We suspect that terminal manufacturers do not follow the recommendation to avoid false negatives that would lead to the rejection of valid transactions.

In all of our experiments, we encountered only one payment terminal type (used exclusively in public transit) where inducing our early CDA failure resulted in an offline decline of the transaction. When inspecting the logs for transactions with this terminal type (not included in Table 1), we noticed that even after being supplied with an invalid CA PK index, the terminal still requests Combined Dynamic Data Authentication (CDA) but aborts the contactless transaction with an error message on the terminal saying that the card is not valid. A possible explanation for this behavior is that the Public Key operations are postponed for efficiency reasons, i.e. the terminal lets the card proceed with the CDA flow before checking the correctness of the certificates. This is, however, not specified in the available documentation.

Our attack did not succeed for the Maestro card used in Transaction 9 of Table 1. We tried both variants of our attack: the downgrade of CVM from PIN to signature and the complete removal of the CVM list (the signature is a valid CVM for this card). In both cases, the contactless transaction was refused by the terminal. The terminal requested the payment to be processed using the regular card reader and then requested the PIN. The logs for these transactions show no major differences when compared with the logs of the transactions where the PIN was successfully bypassed. In particular, neither of the two CVM-relavant data objects supplied by the terminal, i.e. the Cardholder Verification Method Results (CVMR) with value `3F0000` and the Terminal Verification Results (TVR) with value `2400008001`, indicate a cardholder verification failure. Namely, bit 8 of the TVR's third byte was not set, implying that cardholder verification was successful, and the CVMR's third byte was not set to `01`, which would indicate a failed cardholder verification from the terminal. This therefore suggests that the fraud detection system on the issuer's side instructed the terminal in the Online Transaction Authorization (OTA) response to use the contact chip reader.

Discussions with security engineers and architects from the card's issuer confirmed our suspicions. After recent publications of PIN bypass attacks, this issuer has fine-tuned its fraud detection system accordingly and now requests to switch to the contact chip method if irregularities are detected in the contactless transaction. The issuer did not reveal though what finally triggered their fraud detection system to reject our attempted transactions. However, an obvious irregularity is that the 'CDA Failed' bit is set in the Terminal Verification Results (TVR).

We have conducted our experiments with terminals from multiple manufacturers. Except for the case where the terminal insisted on CDA completion and aborted the transaction as explained above, for each issuer, the attacks either worked successfully for all of the issuer's cards (that we had access to), or they did not work for any of the issuer's cards, most probably due to the issuer's fraud detection system.

We close with some observations about the difficulties of live testing. Given the number of parties involved and the opacity of their systems, the only compelling evidence that attacks, similar to the ones described in this paper, really work are live tests. Live tests, however, may raise legal question, even for experiments like the ones we conducted where nobody was defrauded. Except for our open discussion with Mastercard and inofficial exchanges with a major card issuer, the stakeholders in the payment ecosystem seem rather inaccessible. We have suffered from three locked-down accounts at a global terminal provider, most probably due to the unsuccessful transactions associated with our accounts when we experimented with their terminals (when we were tuning our app). Furthermore, this terminal provider seems to have locked some of our cards, since they do not work on this type of terminal even for regular payments, but work instead on the terminals of other manufacturers.

3.5 Countermeasures

The attack described in this paper follows from a combination of flaws that include the improper design of authentication failure modes and user interface weaknesses present in smartphone payments. In this section, we discuss why the fixes proposed in previous work do not defend against the attack presented here. Afterwards, we present our own fixes and comment on deployment efforts.

In our recent work [5], we reported a PIN bypass on Mastercard and related kernels and proposed countermeasures, which we verified using the Tamarin verification tool [22]. These countermeasures require extending the inputs of cryptographic functions, do not affect the EMV execution flow, and rely on offline cryptographic checks. While offering strong cryptographic guarantees, these countermeasures do not prevent the attack we presented here because our attack precisely exploits those executions where the offline cryptographic checks are not performed.

Defending against our attack is nevertheless straightforward, at least conceptually: *transactions where Offline Data Authentication (ODA) has failed must* ***not*** *be accepted*. This can be achieved by at least four different actors that partake in the transaction. Next we list four concrete countermeasures that these actors can implement, where any single countermeasure is alone sufficient to prevent our attack:

C1 The card must always check for consistency between its CVM List and the Cardholder Verification Method Results (CVMR) supplied by the terminal. Namely, the CVMR's first two bytes must be equal to a pair (CVM, condition) in the card's CVM List (the encoding of this list is described in Section 10.5 of [13]).

C2 The terminal must always have the 'CDA Failed' bit of the Terminal Action Code (TAC)-Denial set.

C3 The card issuer (or bank) must always decline transactions where the 'CDA Failed' bit is set in the Terminal Verification Results (TVR).

C4 The payment network (e.g. Mastercard) must always decline transactions where the 'CDA Failed' bit is set in the Terminal Verification Results (TVR).

This list is given in descending order, ordered by the number of worldwide instances of the underlined party. For example, there are billions of cards, but only thousands of card issuers, thus C3 is ordered after C1. It seems reasonable to assume that the implementation costs and deployment efforts for the respective countermeasures are also decreasing with respect to the list above. In line with this, we note that Mastercard has deployed countermeasures, at their network level, to the attack we reported last year [5], which would have taken several months, if not years, if fixes were instead deployed at the protocol level (i.e. on cards and terminals).

Countermeasure C1 would (likely) not work if the card legitimately supports the Paper Signature method, regardless of whether it is the preferred method or not. For example, the card used for transactions 4 and 5 of Table 1 has a CVM List that indicates Paper Signature as a fallback option to Online PIN. Thus, C1 would not prevent our attack for this card. Hence, the countermeasures C2-C4 are preferred.

Countermeasure C2 corresponds to the recommendation in the EMV specification [13] (p. 115) that we discussed in Section 3.4. If CDA has failed, the countermeasure would make the formula in (1) equal to a non-zero value, which consequently would trigger an (offline) decline. As we have already pointed out, according to our observations, TAC-Denial was zero for every terminal we tested with, including those for the transactions in Table 1.

Note that countermeasures C2-C4 require that the card and terminal are both capable of performing Combined Dynamic Data Authentication (CDA). C1 is thus the only countermeasure applicable in cases where neither party (card or terminal) supports CDA.

3.6 Formal Analysis

In our previous work [6], we developed a formal model of the Mastercard contactless protocol and analyzed it using the Tamarin prover [22]. In this model, the rule formalizing the terminal's behaviour, upon receipt of the records sent by the card, requires a valid signature over the received records. The formalization thus models a terminal that would decline any transaction where ODA fails.

The execution trace for the attack that we present in this paper is therefore not contained in the execution space of our previous model [6]. In the following, we therefore explain how this model can be adopted such that Tamarin finds our current attack. Moreover, we discuss how the correctness of the countermeasures just presented can be proven in this model using Tamarin.

Finding Our Attack Using a Formal Model

To account for the corner case in the execution flow where the terminal does not validate the card's certificate and skips integrity checks of the card records, we slightly modify our previous model [6]. Our most significant modification is to add a rule that allows the terminal, after having received the card records, to transition into a state where it is ready to proceed with the GENERATE AC command and afterwards cardholder verification. Our additional rule models exactly the flow described in the specification, where ODA is skipped in cases where the terminal cannot find the CA's certificate indicated by the card.

Additionally, we have implemented several minor modifications to make the new model operational. These include removing the hard-coded restriction that high-value transactions cannot be processed without any CVM.

With our modifications in place, Tamarin finds a valid execution trace for a high-value, PIN-less transaction between a card whose records include Online PIN as a supported CVM and a terminal that does not perform Offline Data Authentication (ODA). The modified model as well as the attack trace found by the tool are available at [2].

Security Proof for Our Countermeasures

Our attack results from an unsafe fallback of a certificate validation error in the Mastercard protocol specification. The fallback tolerates transaction flows with missing integrity protection of critical data exchanged between the card and the terminal. The effects of our four countermeasures are conceptually equivalent: transactions where ODA failed must not be accepted.

The proofs derived from our original model [6] thus directly verify the correctness of our countermeasures. Namely, if the protocol runs without ODA failures, the security properties proven in [6] hold. For our purposes, the most relevant property (see [6], Table II, p. 11) is:

> *The Mastercard protocol with CDA guarantees card and transaction data authentication to payment terminals and card issuers.*

This property guarantees the integrity of critical card and transaction data including the (strongest) CVM supported by the card. This makes our attack infeasible since the card's CVM support cannot be downgraded or removed.

4 Related Work

The widespread use of the EMV payment standard and the devastating impact of vulnerabilities in the standard itself or implementations thereof has attracted the interest of security researchers since its launch. In this section, we review some of the related work on attacks against EMV and comment on their relation to our work. Table 2 summarizes these attacks and their underlying security flaws.

The earliest PIN bypass attack against the EMV payment standard was reported by Murdoch *et al.* [23]. The authors showed that, for transactions with offline PIN verification (i.e. the correctness of the entered PIN is verified by the card), a man-in-the-middle attacker can send a "PIN verified" response for any entered PIN, thereby tricking the terminal into accepting any input as correct.

The underlying flaw that leads to Murdoch *et al.*'s attack is the lack of integrity protection of critical card data. In [16], Ferradi *et al.* analysed about 7,000 fraudulent transactions and concluded that Murdoch *et al.*'s PIN bypass was likely used to defraud 600,000 Euros. In our previous work [6], we observed that Murdoch *et al.*'s attack might still work for some (old) cards that neither support online PIN verification nor have integrity protection for the CVM List (e.g. through the card's PK certificate).

Barisani *et al.* [4] used a card skimmer to attack the EMV protocol for contact cards. Similarly to our own attack presented here, the authors showed that the CVM List could be imperceptibly modified by setting the IAC-Denial to zero to make the terminal accept transactions where ODA failed. In contrast to our attack, Barisani *et al.*'s attack does not induce ODA failures and thus their attack fails for contactless transactions, according to our experiments and our report in [6].

Barisani *et al.*'s attack downgrades the CVM List to Plaintext PIN-only, which then allowed the skimmer device to intercept the PIN sent for verification in plain-text to the ICC. Even if ODA failures are tolerated, this attack can still be prevented by setting the Terminal Action Code (TAC)-Denial object to a specific, non-zero value as recommended in the EMV specification [13] (p. 115). Indeed, the IAC and the TAC are used to decide whether to decline transactions with ODA failures (recall this from Figure 3). Note that this is one of our own countermeasures, as described in Section 3.5.

For VISA contactless cards, Emms *et al.* reported in [12] that cards issued in the UK do not require PIN verification for foreign-currency transactions. The authors built an Android app that performs VISA transactions with cards in NFC range and stores the data, which the authors claim can later be used to request the funds by a rogue merchant terminal. We observe that the now-deprecated *Mag-stripe* mode is used during these artificial transactions and so this attack no longer works.

Although the NFC range is normally limited to a few centimeters, it can be extended to a much larger range. Namely, an attacker can use a pair of devices to interconnect a victim's card with a distant terminal. Numerous attacks of this type, called *relay attacks*, have previously been reported.

The relay attacks reported employ Android devices [17, 20, 27], or customized hardware and software [10, 26], or combinations thereof [7, 9]. Apparently, this type of attack is not lucrative for criminals since it only allows purchases with low amounts; large-value purchases are (presumably) protected

Reported in	**Attack**	**Main flaw(s) exploited**	**Demo'ed in practice**
Murdoch *et al.* [23]*	PIN bypass for contact transactions	No auth. of the card's response to offline PIN verification requests	Yes
Barisani *et al.* [4]*	PIN harvest for contact transactions	Weak auth. of the CVM List	No
Roland & Langer [25]*	Card cloning via legacy modes	Magstripe mode's small pool of random numbers	Yes
[7,9,10,17,20,26,27]	(Passive) Relay	No relay protection	Yes
Emms *et al.* [12]*	Harvest of large PIN-less VISA transactions	No PIN required for foreign-currency contactless transactions	No
Bond *et al.* [8]*	Card cloning via pre-play	Weak random generators	No
Galloway & Yunusov [19]	PIN bypass for VISA cards via relay + MITM	No auth. of CVM-related data sent by terminal and card	Yes
Basin *et al.* [6]	PIN bypass for VISA cards via relay + MITM	No auth. of CVM-related data sent by card	Yes
Basin *et al.* [5]*	PIN bypass for Mastercard cards via relay + MITM	No auth. of AIDs + ambiguity in payment network selection + flaws exposed in [6]	Yes
Radu *et al.* [24]	Apple Pay lock bypass via relay + MITM	Express transit mode + no auth. of CVM-related data sent by terminal and card	Yes
This paper	PIN bypass for Mastercard cards via relay + MITM	Improperly designed ODA failure modes + UI weaknesses in digital wallets	Yes

* attack patched or affected protocol version no longer in use

Table 2: Summary of reported attacks for the EMV payment system

by the card's secret PIN. However, in combination with a PIN bypass attack, as presented in this paper, relay attacks may become worthwhile for criminals. For example, one might imagine a malicious merchant who runs a modified terminal that actually implements a relay in combination with a PIN bypass attack to issue a high-value payment at a different terminal.

In [19], Galloway and Yunusov showed how to use a Rasperry Pi device as an NFC proxy to explore the EMV protocol. In addition, by modifying the 'CVM Required' bit of the terminal's Terminal Transaction Qualifiers (TTQ) and the 'ODCVM Performed' bit of the corresponding card's response, the authors demonstrated the first PIN bypass attack for modern EMV contactless cards.

Another type of attack on EMV is card cloning. For example, back in 2008, Drimer *et al.* [11] showed that the data necessary to clone an EMV card's magnetic stripe can be extracted from an EMV chip using the contact interface. Similarly, Galloway provided a proof-of-concept implementation [18] that proves that this issue still persisted in 2020. Galloway managed to extract all the magnetic stripe data from the card using an NFC reader.

In our previous work [6], we presented a formal model of the EMV protocol. Using our model and the Tamarin verifier [22], we identified a number of weaknesses in the protocol that lead to various attacks, including a PIN bypass attack for VISA contactless cards. This attack is similar but subtly different to the one reported by Galloway *et al.* in [19]: our attack does *not* modify terminal-sourced data (e.g. the TTQ) to avoid potential declines as a result of online integrity check failures. The execution representing the attack that we currently present in this paper is not included in the execution space of our formal model [6] because this model assumes that cryptographic failures result in the execution being aborted, as it is typically the case for formal verification approaches.

In our follow-up paper [5], we presented our *card brand mixup* attack, which allows an attacker to masquerade non-VISA cards as VISA cards. As a result, a payment initiated with a Mastercard-branded card is processed using the VISA kernel in the terminal such that the attack we reported in [6] also applies to Mastercard cards. Our brand mixup attack was patched by Mastercard quickly after disclosure.

In our attack reported here, we use a weaker form of our brand mixup attack on Maestro cards. Concretely, we masquerade Maestro cards as Mastercard cards (see the Steps S1 and S2 in Section 3.2). The terminal-card interaction is the same for all Mastercard-branded cards (including Maestro), but masquerading allows for our CVM downgrade attack, which otherwise is not possible when the payment source is a Maestro card as we observed in our early experiments. We did not find any documentation that would explain differences in CVM processing between Mastercard and Maestro cards.

The most recent attack on EMV is that of Radu *et al.* [24]. The authors have shown that it is possible to unlock Apple

Pay on a (stolen) locked iPhone to make a large purchase. Their attack relies on the works [6, 19], with some adaptations that are specific to the communication protocol layer for smartphone-based payments. This work also analyses the impact of relay-resistance protocols (also known as *distance bounding protocols*) to prevent man-in-the-middle attacks that relay messages. Their analysis is conducted using an extension of our formal model presented in [21].

5 Conclusions

We have reported on a novel, critical design flaw in the EMV protocol for Mastercard contactless transactions. When analysing the Mastercard execution flow, we noted that the online transaction authorization serves as a fallback to offline data authentication failures. We have consequently identified an execution where an early offline data authentication failure makes the terminal ignore all subsequent failures on the PKI-based RSA verification of the card's static and dynamic data during the transaction. Hence we conclude that all card-sourced data that is not authenticated online by the card issuer can be subjected to adversarial modification.

We have developed a proof-of-concept Android app that implements a man-in-the-middle attack that modifies such unprotected data. The modified data includes PKI-related data, to induce the actual offline authentication failures, and cardholder verification-related data, to suppress the card's supported CVMs or to downgrade them to one that criminals can easily reproduce, such as Paper Signature. Using our app, we bypassed the PIN for various transactions with different Mastercard and Maestro cards on different real-world payment terminals. Our experiments clearly demonstrate that the identified flaws substantially degrade the cardholders' security.

We have recommended concrete actions that can be taken to prevent our attack in the short term and we have disclosed them to Mastercard. In the long term, we recommend thoroughly reviewing the input data to all cryptographic constructions of the EMV protocol. We also note that the protocol does not specify terminal authentication mechanisms to cards, which makes the latter communicate with any NFC-enabled device that activates them. This is a shortcoming that undermines the security of the entire EMV system and should also be revised.

EMV Acronyms

AC Application Cryptogram. 6

ADF Application Definition File. 4

AFL Application File Locator. 4

AID Application Identifier. 3, 4, 7, 8, 12

AIP Application Interchange Profile. 4, 6, 7

APDUs Application Protocol Data Units. 3

ATC Application Transaction Counter. 6

CDA Combined Dynamic Data Authentication. 5–11

CDOL Card Risk Management Data Object List. 4, 6

CID Cryptogram Information Data. 6

CVM Cardholder Verification Method. 1–4, 6–13

CVMR Cardholder Verification Method Results. 9, 10

DDA Dynamic Data Authentication. 6

DDF Dictionary Definition File. 3

FCI File Control Information. 4

IAC Issuer Action Code. 7, 8, 11

IAD Issuer Application Data. 6

ICC Integrated Circuit Card. 3, 4, 6, 11

ODA Offline Data Authentication. 1, 4, 6–8, 10–12

ODCVM On Device CVM. 4, 7, 12

OTA Online Transaction Authorization. 6, 9

PAN Primary Account Number. 4, 6

PDOL Processing Data Object List. 4, 6

RID Registered Application Provider Identifier. 7

SDAD Signed Dynamic Authentication Data. 6–8

SFI Short File Identifier. 4

TAC Terminal Action Code. 8, 10, 11

TDHC Transaction Data Hash Code. 6

TTQ Terminal Transaction Qualifiers. 12

TVR Terminal Verification Results. 5, 7, 9, 10

UN Unpredictable Number. 5, 6

References

[1] https://emvrace.github.io.

[2] https://github.com/EMVrace/auth-failure-attack.

[3] Host-based card emulation overview. https://developer.android.com/guide/topics/connectivity/nfc/hce. Accessed: August 2020.

[4] Andrea Barisani, Daniele Bianco, Adam Laurie, and Zac Franken. Chip & PIN is definitely broken: Credit Card skimming and PIN harvesting in an EMV world. In *Defcon*, volume 19, 2011.

[5] David A. 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 Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 179–194. USENIX Association, 2021.

[6] David A. Basin, Ralf Sasse, and Jorge Toro-Pozo. The EMV Standard: Break, Fix, Verify. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 1766–1781. IEEE, 2021.

[7] Thomas Bocek, Christian Killer, Christos Tsiaras, and Burkhard Stiller. An NFC relay attack with off-the-shelf hardware and software. In Rémi Badonnel, Robert Koch, Aiko Pras, Martin Drasar, and Burkhard Stiller, editors, *Management and Security in the Age of Hyperconnectivity - 10th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2016, Munich, Germany, June 20-23, 2016, Proceedings*, volume 9701 of *Lecture Notes in Computer Science*, pages 71–83. Springer, 2016.

[8] Mike Bond, Omar Choudary, Steven J. Murdoch, Sergei P. Skorobogatov, and Ross J. Anderson. Chip and skim: Cloning EMV cards with the pre-play attack. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 49–64, 2014.

[9] 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 - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, pages 189–206, 2015.

[10] Saar Drimer and Steven J. Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*, 2007.

[11] Saar Drimer, Steven J. Murdoch, and Ross J. Anderson. Thinking inside the box: System-level failures of tamper proofing. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 281–295. IEEE Computer Society, 2008.

[12] Martin Emms, Budi Arief, Leo Freitas, Joseph Hannon, and Aad P. A. van Moorsel. Harvesting high value foreign currency transactions from EMV contactless credit cards without the PIN. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 716–726, 2014.

[13] EMVCo. EMV Integrated Circuit Card Specifications for Payment Systems, Book 3, Application Specification, Version 4.3. https://www.emvco.com/wp-content/uploads/documents/EMV_v4.3_Book_3_Application_Specification_20120607062110791.pdf, November 2011.

[14] EMVCo. EMV Contactless Specifications for Payment Systems, Book A, Architecture and General Requirements, Version 2.10. Link, March 2021.

[15] EMVCo. EMV Contactless Specifications for Payment Systems, Book C-2, Kernel 2 Specification, Version 2.10. https://www.emvco.com/wp-content/uploads/documents/C-2-Kernel-2-v2.10.pdf, March 2021.

[16] Houda Ferradi, Rémi Géraud, David Naccache, and Assia Tria. When organized crime applies academic results: a forensic analysis of an in-card listening device. *J. Cryptographic Engineering*, 6(1):49–59, 2016.

[17] Lishoy Francis, Gerhard P. Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical relay attack on contactless transactions by using NFC mobile phones. *IACR Cryptology ePrint Archive*, 2011:618, 2011.

[18] Leigh-Anne Galloway. It only takes a minute to clone a credit card, thanks to a 50-year-old problem. Link, 2020.

[19] Leigh-Anne Galloway and Tim Yunusov. First contact: New vulnerabilities in contactless payments. In *Black Hat Europe 2019*, 2019.

[20] Eddie Lee. NFC hacking: The easy way. In *Defcon*, volume 20, pages 63–74, 2012.

[21] Sjouke Mauw, Zach Smith, Jorge Toro-Pozo, and Rolando Trujillo-Rasua. Distance-bounding protocols: Verification without time and location. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 549–566, 2018.

[22] Simon Meier, Benedikt Schmidt, Cas Cremers, and David A. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pages 696–701, 2013.

[23] Steven J. Murdoch, Saar Drimer, Ross J. Anderson, and Mike Bond. Chip and PIN is broken. In *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA*, pages 433–446, 2010.

[24] Andreea-Ina Radu, Tom Chothia, Christopher J.P. Newton, Ioana Boureanu, and Liqun Chen. Practical EMV relay protection. In *43rd IEEE Symposium on Security and Privacy (S&P)*, 2022.

[25] 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, Washington, D.C., USA, August 13, 2013*, 2013.

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

[27] Jordi van den Breekel. Relaying EMV contactless transactions using off-the-shelf Android devices. In *BlackHat Asia, Singapore*, 2015.