# More is Less: Extra Features in Contactless Payments Break Security

George Pavlides
*Surrey Centre for Cyber Security, University of Surrey*

Anna Clee
*University of Birmingham*

Ioana Boureanu
*Surrey Centre for Cyber Security, University of Surrey*

Tom Chothia
*University of Birmingham*

## Abstract

The EMV contactless payment system has many independent parties: payment providers, terminal companies, smartphone companies, banks and regulators. EMVCo publishes a 15 book specification that these companies use to operate together. However, many of these parties have independently added additional features, such as Square restricting offline readers to phone transactions only, Apple, Google and Samsung implementing transit modes and Visa and Mastercard complying with regional regulations on high value contactless payments. We investigate these features, and find that these parties have been independently retrofitting and overloading the core EMV specification. Subtle interactions and mismatches between the different companies' additions lead to a range of vulnerabilities, making it possible to bypass restrictions to smartphone only payments, make unauthenticated high value transactions offline, and use a cloned card to make a £25000 transaction offline. To find fixes, we build formal models of the EMV protocol with the new features we investigated and test different possible solutions. We have engaged with EMV stakeholders and worked with the company Square to implement these fixes.

## 1 Introduction

No other payment network comes near the scale or processing ability of EMV[1]. Its standardisation body, EMVCo, has issued EMV specifications stretching across 15 books[2]. In the last three years the market of payment readers doubled in size to $33.5 billion[3] with providers such as Square and SumUp creating new, portable EMV terminals for the general public. EMV companies have added a range of new features, full details of which are only found in private, proprietary documents available only to those with operating licences, if at all. Recent attacks, reverse-engineering work and formal analyses (e.g., [40, 42, 44, 45, 52]) of EMV uncovered some details of these closed EMV specifications, and show that, as features are added, the number of attack vectors increases.

We look at how the existing EMV framework has been used to allow for new features that are not fully in the EMV specifications, but rather retroactively fitted in or overloaded. By looking at an offline terminal, we are also able to distinguish between checks made by the terminal and the backend. Through testing to understand their behaviour, building formal models to include this found behaviour and checking how these extra features interact with one another, we find numerous attacks and flaws caused by these extra features.

**Extra Features: Cardholder verification methods, Transit, Tap and PIN, and Offline Readers** Mobile-phone payments have replaced PIN authentication with a Consumer Device Cardholder Verification Method (CDCVM) that uses a fingerprint or faceID. Visa define their implementation in the EMV specification, with a flag in the protocol that indicates if CDCVM has been performed. Mastercard does not include details of their implementation in the EVM specification. Past work found that Mastercard checked the CDCVM status using an undocumented proprietary field in the protocol [52]. We find that this is a backend check by Mastercard, and hence this security check is absent for offline terminals, which contributes to our attacks.

Transit mode payments have been designed by Apple, Google and Samsung as a fast way to pay low values on transport systems without the delay of unlocking a phone. How different phones detect transit readers is not publicly documented. Past work found that Apple use a system called Apple Enhanced Contactless Polling [52]. We look at how this feature is implemented on GooglePay and find that, for Visa, it looks for a field in the Terminal Transaction Qualifiers (TTQ) that indicates that the terminal supports "offline authentication for online authorizations". The EMV specification defines this as a general flag that can be used by any reader that may go offline. We find that some Square shop terminals use this flag as defined in the specification. So Google's decision to overload the meaning of this field as a way to detect transit readers means that GooglePay can be used on offline Square readers without any customer authentication, bypassing the need to unlock the screen.

Regulators in certain countries, such as the UK Payment Services[4] require that cards are inserted into the terminal for PIN verification. Meanwhile, in most of the EU, *"Tap-and-PIN"* allows contactless plastic cards to use a PIN. The EMV specification does not explain how "Tap-and-PIN" and "non-Tap-and-PIN" differ. We uncover how Visa and Mastercard have implemented these regulations, finding that Mastercard uses an undocumented, backend check, and that it is possible to bypass these regulations on Visa, performing a Tap-and-PIN transaction with a non Tap-and-PIN card.

Offline capable terminals can accept EMV payments when not connected to the Internet, and later, when the terminal reconnects, the payment information is sent to the bank for approval. Due to different business rules in different countries, whether plastic cards are allowed in contactless offline transactions varies. For the US *Square Terminal* this is allowed [5], but in the UK only mobile wallets are allowed during offline transactions. This might help reduce fraud because a mobile wallet can perform *CDCVM*, authenticating the user. We investigate how Square restricts terminals to mobile wallets, and find that this is based on detecting the presence of the proprietary Value Added Services (VAS) loyalty card system, which is implemented on smart phones. We show that this can be bypassed to get plastic cards accepted offline, by replaying a phone's response to VAS. Combined with the undocumented way in which Mastercard checks for CDCVM, this makes it possible to perform an over-the-limit (OTL) attack against offline Mastercard, which is rejected later when the terminal goes online (after a thief may have left a shop with their fraudulent purchase).

The above features and regulations are not considered in the EMV specifications. By relying on retrofitting and overloading for accommodating features and regulations, EMV stakeholders open new attack vectors without understanding the overall effect on the whole system. We look at a handful of these extra features, and find numerous attacks, showing this is a systemic oversight. The use of ad-hoc, proprietary extensions means that EMV loses the security it gained through carefully designing the protocols in the core specification.

We proposed fixes for the attacks we found by considering all the features from the different companies together, which we did using a formal model. The use of formal models, checked with the Tamarin prover, gives us some confidence that we have avoided the kind of design mistakes we found in the features. Building formal models forces us to write a rule for every message, which ensures that we have considered how every field in every message is processed. While we did not discover the attacks presented here as a result of running the Tamarin prover, we did spot many of the attacks while building the formal models due to the rigorous, complete consideration of the protocol that this requires. There are no public details on the backend checks made on EMV transactions, so we used the formal model to help us reverse engineer the backend system by testing different possible checks the backend could make and seeing what is consistent with the behaviour we observed from terminals. Square has now implemented our recommended fixes.

**Contributions:**

- We investigate and explain how a range of out of specification EMV features work.
- We find a range of attacks against these features.
- We show that these flaws are a systemic issue often due to interactions between undocumented ad-hoc features.
- We suggest and formally verify protocol fixes, some of these have already been implemented by Square.

## 2 Background & Related Work

We now present main EMV protocols and features, with emphasis on online vs. offline EMV, and customer validation. Appendix G contains a list of EMV acronyms.

### 2.1 EMV Parties & Online/Offline Executions

The EMV protocols are payment protocols designed to secure and authenticate the transaction, but also to keep everything interoperable, allowing for many variations of set up. There are four main parties: *payment devices* which can be traditional bank-cards, that we refer to as *plastic cards*, and *mobile devices* which are phones, smartwatches, etc. running the protocol via a mobile-payment app such as ApplePay, GooglePay or SamsungPay with a card or several registered on this app. We will refer to these generally as the payment device. An essential difference between the two types is that mobile devices have *Consumer Device Cardholder Verification Method(s) (CDCVM)* (e.g., via a PIN, a fingerprint or FaceID verified by the device), whereas this verification of cardholders cannot be done by plastic cards themselves, so instead can be performed by the next party, the reader. *Points of Sale (PoS)*, referred to as *(EMV) readers* or *terminals* can perform authentication of the cardholder by requesting a PIN be inputted to the reader. Then, there are the banks that issue the cards to be used via the payment device; we refer to these simply as the `Bank`[6]. Finally, there are the *payment networks*, represented by the card providers, e.g., Visa, Mastercard.

Every card has cryptographic data assigned to it upon issuance, which proves it is a valid card from its `Bank`. This includes a symmetric key $K_M$ shared with the `Bank`, a private-public key pair and a bank-signed digital certificate on the public key. This public key is retrieved by the terminal during the protocol execution, whichever the protocol type is (Visa, Mastercard, etc.). During the protocol, the payment device packages into what is called an *Application Cryptogram (AC)* its view of the transaction. This includes a field called the *Issuer Application Data (IAD)*, which has proprietary descriptions consumed by the bank that issued the card and the payment network. The IAD is also a backend tool that describes[7] which type of payment device did the transaction, in which conditions (e.g., transit), if CDCVM was used, etc. The Application Cryptogram (AC) is a MAC that only the bank can verify, and is used when the bank determines whether to approve a transaction. The terminal can be either: *online* (i.e., connected to the Internet), *offline* (i.e., disconnected from the Internet), or –as some PoS onboard transport vehicles would be – *offline for online* (i.e., likely having an Internet connection, but performing a transaction as if they were offline just in case they lose connectivity). During the protocol, the terminal sends to the payment device its online/offline nature.

Offline PoS. If the terminal is offline, elements of the transaction as seen by the payment device are packaged by the payment device into the *Signed Dynamic Application Data (SDAD)* – a signature by the card which the terminal can verify. The payment device will send the *SDAD* and the *AC* to the terminal for both the Visa and the Mastercard protocols. The offline terminal must verify this *SDAD* and other data sent by the payment device during the protocol (e.g., expiry date) and accept or decline the payment based on this information. We describe how the certificates and SDAD are decoded in the appendix, and give examples of their contents. This verification step ensures that the payment device's impression of the protocol's execution (e.g., a nonce exchange) is in line with that of the terminal. If the terminal is offline, the protocol finishes between the payment device and the terminal at this point; if the terminal's checks are successful, the cardholder leaves the merchant's premises and be deemed to have paid for goods or services.

When the offline terminal reconnects to the Internet (which may be days after the transactions), the terminal sends the *AC*s of all offline transactions to the corresponding banks, accompanied by data describing the transaction. The bank (and the payment networks) do checks on the *AC* and the terminal information, and examine whether there are funds in the bank account of the card or fraudulent activity linked to it, before authorising/declining the payment.

Online PoS. If the terminal is *online*, for Visa, the payment device will not send it the *SDAD*, but only the *AC*. For both Mastercard and Visa, the *AC* will be forwarded by the terminal to the `Bank` in real time; the online terminal will authorise the payment if the bank and payment network approved it.

"Offline for Online" PoS. Readers with a transport Merchant Category Code (MCC) advertise to be *"offline for online"*. In this mode, readers may be connected to the Internet or not. The payment device sends to these readers the *SDAD* and the *AC*, but the terminal is not mandated to check the *SDAD*, if it is online and can verify the AC with the backend. Depending on the connection and configuration of the reader, from here on, it may behave like an online terminal or like an offline reader: i.e., forward the *AC* to the `Bank` in real time or not, receive payment-authorisation results from the bank or make its own decisions w.r.t. payment authorisation.

## 2.2 On-Device Customer Validation (CDCVM)

EMV transactions with contactless cards remain entirely contactless (needing no cardholder input) given the following spending limits are not exceeded: a limit per transaction (e.g., £100 in the UK) and/or cumulative limits (e.g. £200). If either is exceeded, then the transaction generally requires *Cardholder Verification (CV)* mechanism.

The terminal gets from the payment device information about the *CV Method(s) (CVM(s))* it supports, as well as the conditions in which these rules apply. For plastic cards, the CVM is generally such that the terminal requests the card's Personal Identification Number (PIN) be input into the terminal; for online readers, this PIN is sent inside the AC to the bank for verification, whereas for offline readers it is checked against the card's persistent memory. In contrast to this, on mobile devices, CV can be done by what is called *Consumer Device Cardholder Verification Method (CDCVM)*, e.g., the user's fingerprint or face is scanned by a mobile app. So, in a CVM negotiation between the payment device and the terminal, a contactless payment above the contactless limit may be accepted by a terminal without the terminal doing any CV, as the CV was delegated to the payment device.

## 2.3 The Visa & Mastercard Protocols

**Visa's Protocol.** This protocol, with the EMV fields most relevant to this paper explained, is depicted in Figure 1. First, the payment device (on behalf of the card) and the terminal depicted in the figure as the reader, negotiate if they are to run Visa, Mastercard, etc. via the "SELECT" messages. Then, the data needed for the transaction is requested by the payment device in the Processing Options Data Object List (PDOL) (e.g., amount to pay, a nonce generated by the terminal denoted $UN$, etc.). The terminal sends this data in the Get Processing Options (GPO) message, a field of note in this is the TTQ. The TTQ informs the payment device whether the terminal is online, offline or 'offline for online' and whether a Cardholder Verification Method (CVM) is required for this transaction (e.g., because it is over the contactless limit).

The payment device replies to the GPO message with a series of EMV fields and data, of which we mention: the Card Transaction Qualifiers (CTQ) – a field saying how the payment device can comply with the TTQ requirements (e.g., if it did on-device authentication (CDCVM) or not when the TTQ asked for it); the Application Interchange Profile (AIP) – a field describing the capability of the payment device (e.g., if it is capable of on-device authentication or not, which mobile phones are likely to be, whereas "plastic" cards are not); and a nonce $N_C$ carrying the randomness the payment device adds to the cryptographic data (i.e., the *AC*, *SDAD*) in this particular execution. Next, the terminal requests data from the payment device (RECORDS), which includes the certificates of the card, these are checked by the terminal. The payment device

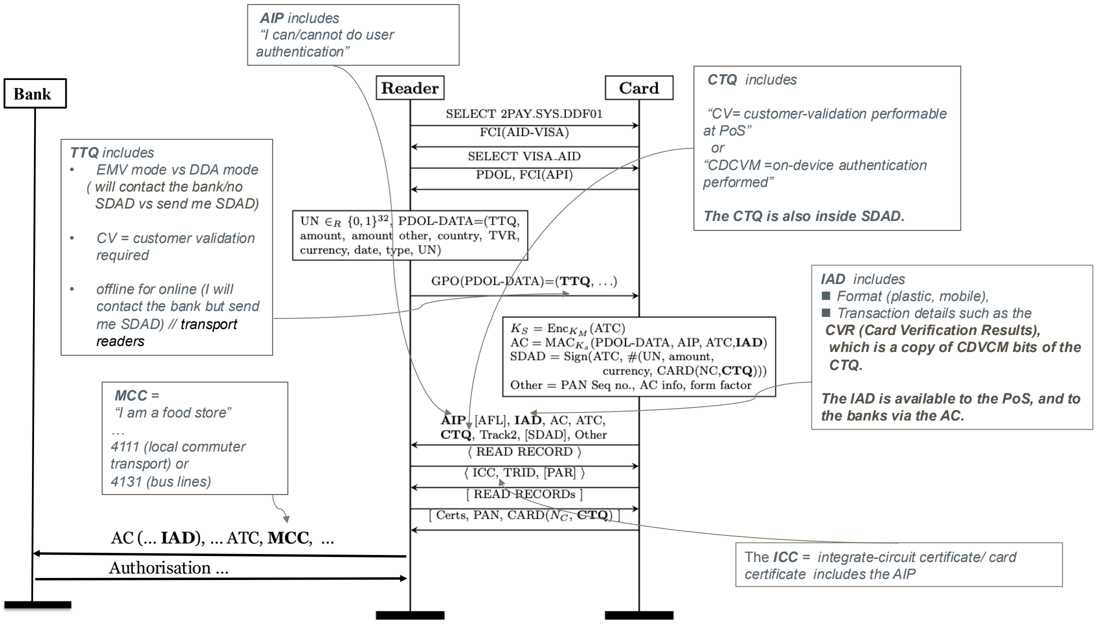
AIP includes
"I can/cannot do user authentication"
Bank
Reader
Card
CTQ includes
"CV= customer-validation performable at PoS"
or
"CDCVM =on-device authentication performed"
The CTQ is also inside SDAD.
SELECT 2PAY.SYS.DDF01
FCI(AID-VISA)
SELECT VISA_AID
PDOL, FCI(API)
TTQ includes
• EMV mode vs DDA mode ( will contact the bank/no SDAD vs send me SDAD)
• CV = customer validation required
• offline for online (I will contact the bank but send me SDAD) // transport readers
$UN \in_R \{0,1\}^{32}$, PDOL-DATA=(TTQ, amount, amount other, country, TVR, currency, date, type, UN)
GPO(PDOL-DATA)=(**TTQ**, ...)
IAD includes
■ Format (plastic, mobile),
■ Transaction details such as the CVR (Card Verification Results), which is a copy of CDVCM bits of the CTQ.
The IAD is available to the PoS, and to the banks via the AC.
$K_S = \mathrm{Enc}_{K_M}(\mathrm{ATC})$
$\mathrm{AC} = \mathrm{MAC}_{K_s}(\text{PDOL-DATA, AIP, ATC,}\mathbf{IAD})$
$\mathrm{SDAD} = \mathrm{Sign}(\mathrm{ATC}, \#(\mathrm{UN}, \mathrm{amount}, \mathrm{currency}, \mathrm{CARD}(\mathrm{NC},\mathbf{CTQ})))$
Other = PAN Seq no., AC info, form factor
MCC =
"I am a food store"
...
4111 (local commuter transport) or
4131 (bus lines)
**AIP**, [AFL], **IAD**, AC, ATC, **CTQ**, Track2, [SDAD], Other
⟨ READ RECORD ⟩
⟨ ICC, TRID, [PAR] ⟩
[ READ RECORDs ]
[ Certs, PAN, CARD($N_C$, **CTQ**) ]
AC (... **IAD**), ... ATC, **MCC**, ...
Authorisation ...
The ICC = integrate-circuit certificate/ card certificate includes the AIP

Figure 1: Visa's PayWave Protocol. (Most-relevant EMV fields in bold face and explained; optional messages in brackets).

sends the requested data including the AC and, if offline or offline for online, the SDAD too. The AC contains lots of transaction data to depict the payment device's view of the exchange. See Fig. 1 for all that is contained in the AC. Worth highlighting is the IAD: copies of parts of the CTQ (notably a field called Card Verification Results (CVR) which says if the payment device performed on-device authentication or not) appear in a part of the AC called the IAD, which the banks should check. We decoded SDADs (see Appendix D), and found that the SDAD contains the Card Authentication Related Data (CARD), holding the card's nonce and a copy of the CTQ. This should be verified, *offline*, by the terminal.

Finally, the terminal sends via the secure channel the details of the transaction to the bank and payment networks (e.g., Visa). This includes but is not limited to details of the terminal e.g., Merchant Location, *Merchant Category Code (MCC)* denoting if the terminal is a retail shop or a transport terminal (e.g., MCC 5732 is for electronics stores, 4111 is for local transport), etc. for anti-fraud checks and fee charging [56].

**Mastercard's Protocol.** We diagrammatically give Mastercard's protocol, called PayPass, in Figure 8 (in Appendix F, due to space), detailing particularly the elements most pertinent here, and differences from the Visa protocol. Some of the EMV fields Mastercard has are in common with Visa's protocol (e.g., Application File Locator (AFL) – the index of all the static "records" available on the payment device, the AIP – the payment device's capabilities), but these fields are consumed in a different order to Visa's protocol. There are also differences to Visa. For instance, the terminal uses the AFL to request what are called "Track 2" (the user's account information), and the CVM List. The latter is comparable in purpose to the CTQ in Visa, declaring payment device's different CV abilities.

The terminal will pick one of the CVMs advertised by the payment device and return its choice to the payment device as the *CVM Result*. The SDAD in Mastercard contains more EMV Fields than in Visa: e.g., the SDAD now includes the full Card Risk Management Data Object List 1 (CDOL1) data and the AC. For Mastercard, the AIP, which indicates the payment device's capabilities, is inside the authenticated Integrated Circuit Card Certificate (ICCc). Mastercard-compliant readers always check the certificates and the SDAD, and if these are correct, they send (when online) the payment device-issued AC, containing the payment device-issued IAD and its own transaction data to the issuing bank and the payment network. The bank will verify the AC, and the payment network (e.g, Visa, Mastercard) will verify the IAD. If these are correct, the payment will be authorised (online).

**Transit Mode for Mobile Contactless EMV.** "Transit mode" is a way in which mobile payment devices and applications (e.g., ApplePay, SamsungPay, GooglePay) can operate during contactless EMV transactions, without complying to standard CDCVM rules, in order to increase user-friendliness. To facilitate usability when passing through a transport barrier, e.g., in the London Underground, the CDCVM requirement is bypassed when the mobile payment device identifies it is transacting with a "transport terminal". The mobile device allows a payment *without the device being unlocked*, and without the user performing CDCVM validation, i.e., input their payment app PIN, present their FaceID/fingerprint.

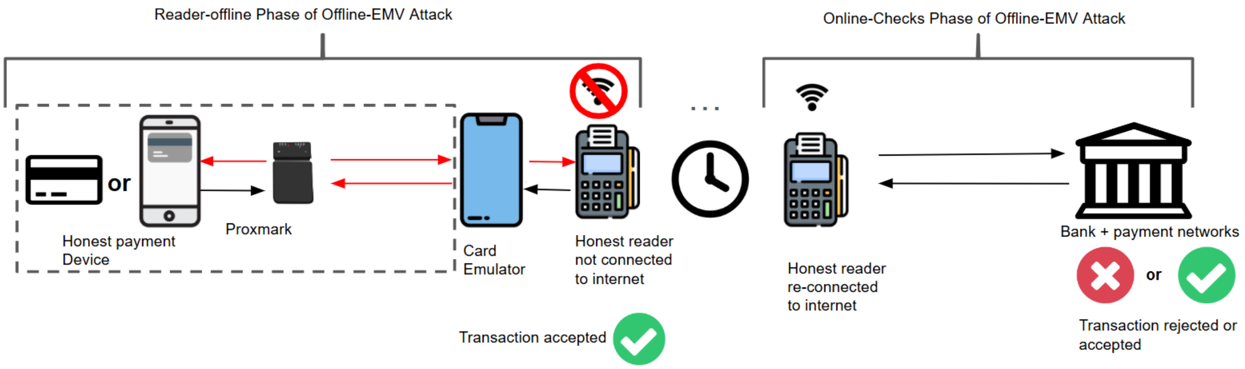
Reader-offline Phase of Offline-EMV Attack
Online-Checks Phase of Offline-EMV Attack
or
Honest payment Device
Proxmark
Card Emulator
Honest reader not connected to internet
Transaction accepted
Honest reader re-connected to internet
Bank + payment networks
or
Transaction rejected or accepted

Figure 2: Our Methodology for Offline-EMV Attacks (arrows form a relay-based MiM attack; red arrows carry active-MiM changes; parts in the dashed box – optional)

## 2.4 Related Work

Galloway et al. [49] were the first to bypass the contactless-limit authentication by setting the "CVM required bit" in the TTQ to 0, and the "CDCVM bit" in the CTQ to 1; this was done against UK cards. A Galloway-like attack was also performed on Swiss contactless cards by [42]; business rules differentiating UK cards from Swiss ones were essential, but this was not known at the time; Radu et al. [52] realised it and dubbed the Swiss behaviour "Tap-and-PIN". We finally uncover what this in EMV is and how this works.

Yunusov et al. [57] [55] extended the Galloway attack, by using a compromised terminal, which is out of our scope. Radu et al. [52] also extended the Galloway attack to make an iPhone think it is in transit mode and thus incorrectly does payments without user-authentication. They also described details of the proprietary IAD. For Mastercard, the reader also gets an authenticated copy of the IAD inside the SDAD. For both Visa and Mastercard, the IAD is sent by the payment device to the terminal in the clear, and the issuer gets the IAD authenticated inside the AC. So, issuers can reliably check the IAD for its inner CDCVM data and operation modes (e.g., if the payment was performed in transit mode or not, and even if it was done with an ApplePay, a SamsungPay, a GooglePay, etc.), as well as check it against the MCC and other transaction data sent by the payment device and the terminal. This has not occurred in prior CDCVM- or transport-mode based *online* attacks by [42, 49, 52]. We find this is still not the case, even when our attacks modify more EMV data.

Recently, Basin et al. [41] experimented with making an incorrect Mastercard transaction by altering the card-certificate validation process on the terminal. An attack we present against SamsungPay and Square is similar in nature.

There is also a history in the formal analysis of EMV. Formal Dolev-Yao [47] models of EMV in verification tools such as ProVerif [43] or Tamarin [51] include [42, 45, 46, 52]. In this work, we enhance the models of [52] in line with the attacks and countermeasures advanced herein.

# 3 Threat Model & Attacks' Methodology

## 3.1 EMV and Proprietary Regulations

**Regulations on Payment Devices and Readers.** There are many restrictions in payment regulations[8]. Some payment devices do not work with certain readers or in specific conditions (e.g., SamsungPay on some transport readers). And, terminals accept specific cards and devices (e.g., no Amex, no SamsungPay): *e.g., Square's website says that, in Europe, the Square Terminal works offline, contactless, only with the mobile-wallets ApplePay and GooglePay, using CDCVM*[9], *not with plastic cards or other devices/wallets.* Retailers can set a limit for offline payments, of their choice; the *Square Terminal* can be set to a maximum of £25,000 offline.

**Regulations on Terminals, Banks and the Payment Service and networks.** EMV enforces transaction limits for contactless payments without user authentication, varying by region (e.g., £100 in the UK, €150 in the EU). Implementation of EMV beyond this limit differs by country. *In most of Europe, PINs can be entered without inserting the card into the terminal, while in the UK, transactions over these limits require inserting the card, moving from the contactless to the contact protocol. More details are in Section 4.2.*

## 3.2 Threat Model, Attacks' Phases & Victims

We consider, honest payment devices, terminals, banks and payment networks. Corrupting a payment device or bank is considered hard and the EMV system provides no protection against this. While the terminals are built to be tamper-proof they may be corrupted but this can be traced back to the owner[10]. We consider a Dolev-Yao [47], Man-in-the-Middle (MiM) protocol attacker, which can replay and alter messages, while controlling the communication between the card and the terminal, for any possible combination of sessions, but without performing cryptographic attacks against the signature/MACing primitives used in the protocols. The attacker has access to an honest payment device and terminal, and could also have and use a stolen payment device.

We say a transaction is *correct* if all EMV requirements and regulations are observed. An *EMV attack* is a series of actions by our attacker that result in an incorrect EMV transaction being authorised. Working with offline readers, we say that an *offline-EMV attacks* has two *phases*:
1. **reader-offline phase** – the transaction is incorrect but authorised whilst the reader is offline;
2. **online-checks phase** – the transaction is incorrect, and it is *also* authorised when the reader goes online.
*Importantly, both phases of offline-EMV attacks are themselves attacks in their own rights, since payment authorisations take place when they should not.*

There are several dimensions to our attacks: i.e., the victims , the underlying cause (e.g., non-compliance to specification, or even flaw in the specification), if it is EMV-wide or company-specific, etc. All of these are summarised in Figure 5. There are several such dimensions that characterise ours attacks. One full characterisation, which is also backed by our formal models (see Sections 5 and 8) is given by the following three dimensions: 1) breach of regulations (relating to Tap-and-PIN or offline payments); 2) "free lunch" nature (when the transaction is accepted offline, and rejected when the terminal goes online, so the attacker gets a "free lunch" where they have received goods/services without paying, leaving the merchant out of pocket); 3) "over-the-limit" illicit payment, offline or online, (when an un-authorised contactless payment is accepted for any amount, especially over the contactless limit). Each of our attacks falls in one, two or all of these categories.

### 3.3 Our Tools and Methodology

We used a series of well-established RFID and EMV toolkits. We used a Proxmark3 RDv4[11], a programmable toolkit with EMV/RFID antennas, to: (1) sniff our transactions to get so-called "EMV traces", to understand protocols better (e.g., in Section 6); (2) emulate an EMV reader against a victim payment device (Section 4 and Section 6). We built and used Android-based payment device emulators (Section 4 and Section 6). We used both of these, alongside numerous cards (Visa and Mastercard, issued in various countries), mobile-wallets on phones (various OSs and models of Samsung, Nokia, Apple), readers (SumUp Air [12], SumUp Solo[13], Square Reader[14], Square Terminal[15] registered in the UK and in Romania) to do reverse engineering (e.g., Section 4.2, Section 6, Section 7) and mount attacks (Sections 4 and 6).
**Choice of Terminals.** The terminals used for each of the main EMV functionalities investigated are summarised below, along with the motivation behind choosing them.
Offline functionality: Square Terminal (registered in the UK). Square is one of the most widely used terminal providers [16] and Square Terminal is the *only offline-capable* terminal available in the UK without having a registered company.
Tap-and-PIN functionality: SumUp Air (one registered in Romania and one in the UK), SumUp Solo (registered in the UK), Square Reader (registered in the UK) and Square Terminal (registered in the UK). The aforementioned terminals are manufactured by popular terminal companies and are available to the general public without having a registered company. We believe that four readers over two distinct providers and countries are enough to understand and conclude how the Tap-and-PIN functionality is enforced.

Much of this work is similarly to [49, 52], on relay-based MiM. The main difference is that, since we work with offline readers, some of our MiM have two phases; this is shown in Figure 2. This figure also shows that, in some attacks (ATTACK 1-4), we use the full MiM, whereas in others (ATTACK 10), we utilise a Host-based Card Emulation (HCE). A similar figure for comparison for online-EMV attacks, is given in Figure 7 in the appendix.

Note. In all our attacks, the shop assistant will see our phone (payment device emulator) paying at a terminal, so it will all look like an inconspicuous, normal mobile payment.

## 4 Testing Features Centered on Plastic Cards

In this section, we look at how Square stops offline payments from plastic cards and how Visa and Mastercard stop some plastic cards from doing Tap-and-PIN transactions, leading to 4 attacks. Neither of these features is detailed in the EMV specification, so our work is informed by experiments and past work as shown in the mind-map in Figure 3.

### 4.1 Offline restrictions to smart phones

**The Feature:** The first functionality we look at is the *Square Terminal*'s restriction of offline, contactless transactions to only work with mobile phones, blocking plastic cards. We note that, the EMV specification does not define how to differentiate between plastic and mobile payments.

**Our Experiments and Results:** To investigate *Square Terminal*'s distinction, we captured a trace of a successful iPhone transaction and a failed plastic card attempt. The failed transaction revealed the *Square Terminal* sent a "SELECT OSE.VAS.01" message instead of the standard "SELECT 2PAY.SYS.DDF01". We discovered this is part of the *VAS* protocol, which we then explored.

**Analysis:** "Value Added Services[17]" (VAS), also known as "Smart Tap[18]" by Google, is a proprietary protocol for adding loyalty cards to mobile wallets. Official documentation is only available under NDA. However, reverse engineering efforts by Grayson Martin[19] and Kormax[20], and information from SpringCard[21] and ID TECH[22], provide some details. Using these sources, we describe the protocol, with details in Appendix C, though it is not central to our research.

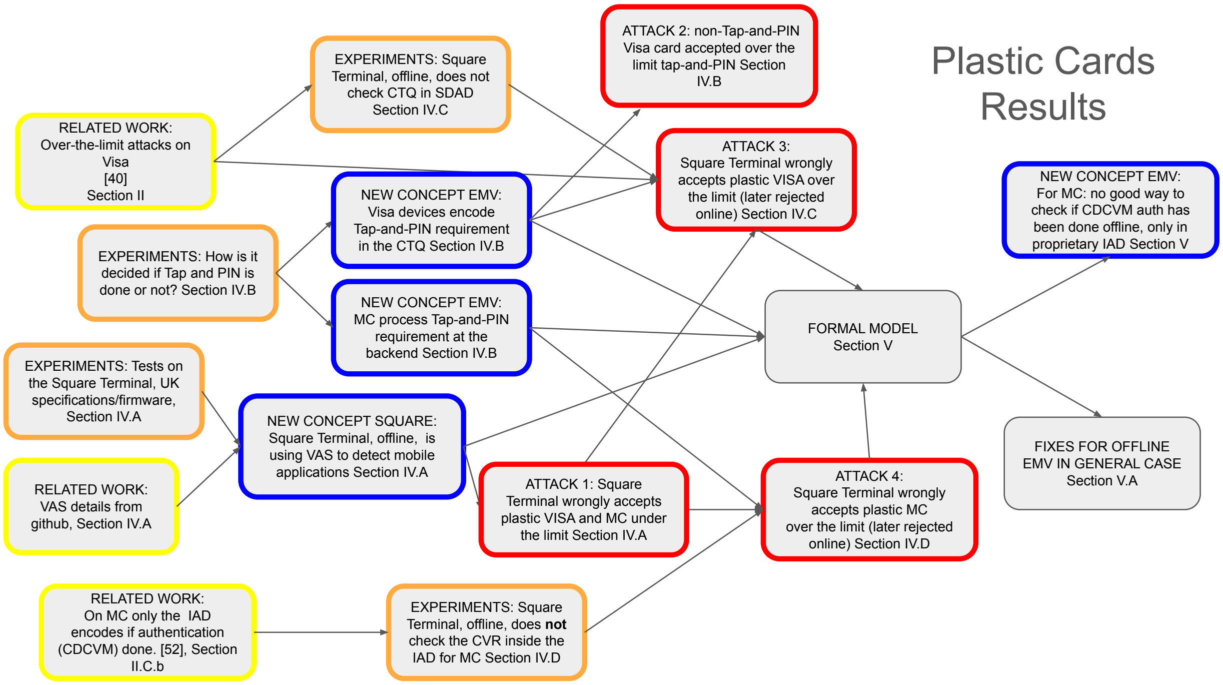
Plastic Cards Results
RELATED WORK: Over-the-limit attacks on Visa [40] Section II
EXPERIMENTS: Square Terminal, offline, does not check CTQ in SDAD Section IV.C
ATTACK 2: non-Tap-and-PIN Visa card accepted over the limit tap-and-PIN Section IV.B
ATTACK 3: Square Terminal wrongly accepts plastic VISA over the limit (later rejected online) Section IV.C
NEW CONCEPT EMV: Visa devices encode Tap-and-PIN requirement in the CTQ Section IV.B
EXPERIMENTS: How is it decided if Tap and PIN is done or not? Section IV.B
NEW CONCEPT EMV: MC process Tap-and-PIN requirement at the backend Section IV.B
NEW CONCEPT EMV: For MC: no good way to check if CDCVM auth has been done offline, only in proprietary IAD Section V
FORMAL MODEL Section V
EXPERIMENTS: Tests on the Square Terminal, UK specifications/firmware, Section IV.A
NEW CONCEPT SQUARE: Square Terminal, offline, is using VAS to detect mobile applications Section IV.A
RELATED WORK: VAS details from github, Section IV.A
ATTACK 1: Square Terminal wrongly accepts plastic VISA and MC under the limit Section IV.A
ATTACK 4: Square Terminal wrongly accepts plastic MC over the limit (later rejected online) Section IV.D
FIXES FOR OFFLINE EMV IN GENERAL CASE Section V.A
RELATED WORK: On MC only the IAD encodes if authentication (CDCVM) done. [52], Section II.C.b
EXPERIMENTS: Square Terminal, offline, does not check the CVR inside the IAD for MC Section IV.D

Figure 3: Mind-map of the Main Results from Sections 4 and 5

The plastic card responded to the Square Terminal's "SELECT OSE.VAS.01" with an ISO 7816 error, indicating the service was unavailable, ending the transaction. In successful iPhone transactions, the phone sent its VAS capabilities. Since the *Square Terminal* did not consume the data sent in VAS, and VAS is only supported on smartphones, we speculated it used "VAS SELECT" to identify mobile wallets.

To test this, we recorded a legitimate VAS response from an iPhone (with a Visa card) and added it to our relay-based MiM attack in Figure 2: concretely, our card-emulator sent this VAS reply to the *Square Terminal*, while we relayed the rest from the plastic card, leading to our first attack:

**ATTACK 1:** We can bypass Square's restriction offline to only mobile devices, by replaying a recorded VAS response, then relaying messages from a plastic card.

**Discussions:** ATTACK 1 succeeds during the "reader-offline phase". When the *Square Terminal* goes online, under-the-limit (UTL) payments are approved, indicating no additional checks to prevent plastic card use.

While not particularly dangerous we rate this as a low impact attack, since plastic cards can make online payments. However, we note that it breaches *Square Terminal* intent to block offline plastic-card transactions in certain regions, necessary to comply with financial regulations[23]; so, *Square Terminal* does not adhere robustly to regulations that are there to mitigate the added risk of offline payments, which typically require user authentication (e.g., CDCVM).

While the past work on reverse engineering VAS helps us understand and parse the messages, Square's use of just the first exchange of VAS messages to detect a smart phone is novel, as is our attack against this.

## 4.2 Tap-and-PIN vs Non-Tap-and-PIN Cards

**The Feature:** Section 2.2 explained that over-the-limit contactless transactions with plastic cards require PIN validation. However, PIN validation for contactless EMV varies geographically due to different rules and protocols. In countries like Spain, France, and Germany, over-the-limit transactions remain contactless, with the PIN entered without card insertion (**"Tap-and-PIN"**, as coined by Radu et al [52]). In the UK and Singapore, over-the-limit transactions require card insertion and PIN entry, known as contact/chip-and-PIN (**"non-Tap-and-PIN"**, as coined by Radu et al [52]).

**Our Experiments:** We aimed to understand and potentially bypass these geographic restrictions. To carry out the experiments, we used Tap-and-PIN cards from Romania, Turkey, Germany and France, and non-Tap-and-PIN cards from the UK. We also used the terminals mentioned in Section 3.3 in online-mode. The reader registered in Romania does Tap-and-PIN and the readers registered in the UK do not do Tap-and-PIN. First, we sniffed and analysed EMV traces produced by combinations of readers and card types.

**Results and Analysis for Mastercard Tap-and-PIN:** We found that all Mastercard cards performed Tap-and-PIN with the Romanian reader, but none did with the UK readers. Inspecting traces showed that the only differences where in fields unrelated to Tap-and-PIN, which we would expect to be different between traces (i.e., the nonce, time, AC & SDAD).

From EMV specification, two EMV fields, *Terminal Verification Results (TVR)* and *Cardholder Verification Method (CVM) results*, could indicate if Tap-and-PIN is required. The TVR field reflects the terminal's transaction status, while the CVM fields shows the verification method used.

Looking at traces, in a Tap-and-PIN transaction, the fact that a PIN is entered is reflected in the TVR and CVM results. However, we experimented with these fields and found that the card does not refuse to complete an over-the-limit transaction, even when the TVR and CVM results indicate no PIN was entered. So, Mastercard does the checks for Tap-and-PIN purely at the backend, and not on the card. This will also be a contributing factor to us being able to mount an *offline-EMV*, over-the-limit attack for Mastercard (i.e., ATTACK 4), discussed later in Section 4.3.

**Results and Analysis for Visa Tap-and-PIN:** Unlike Mastercard, non-Tap-and-PIN Visa cards would not perform high-value Tap-and-PIN transactions with the Romanian EMV reader. Past research suggested non-Tap-and-PIN cards only do contactless transactions below the limit [48], but we found this was false: if the "CVM required" flag is set in TTQ, the cards reply with a 6984 error code ("referenced data reversibly blocked[24]"), rather than providing a payment AC.

We then used our online-MiM method (see Figure 7) to unset the "CVM required" flag. Comparing the traces of the Tap-and-PIN cards and the non-Tap-and-PIN cards (with this "CVM required" flag unset), we found a difference in the CTQ field. The Tap-and-PIN cards set the bit called "Online PIN required" (Byte 1 Bit 8), while the non-Tap-and-PIN cards do not. So, we set our relay to clear the "CVM required" bit inside the TTQ, and also set the "Online PIN required" bit in the CTQ. In this case, the reader asks for a contactless PIN from the "non-Tap-and-PIN" card and completes an over-the-limit transaction as Tap-and-PIN. So, we mount this:

**ATTACK 2:** For Visa and all online EMV terminals, we can perform a Tap-and-PIN transaction with a non-Tap-and-PIN card, by setting "Online PIN required" flag (bit 8 of byte 1) in the CTQ, and unsetting "CVM required" flag (bit 7 in byte 2) in the TTQ.

**Discussions.** ATTACK 2[25] breaks Visa's security model by enabling Tap-and-PIN, though requiring the attacker's knowledge of the PIN makes it less useful. Still, like ATTACK 1, it violates regulations and bypasses card payment restrictions, that issuers must abide to in order to do business.

To the best of our knowledge this is the first work to explain how readers distinguish cards that should and should not do Tap-and-PIN. Our findings also clarify discrepancies in two past over-the-limit Visa attacks which were down to lack of understanding of Tap-and-PIN workings: Yunusov and Galloway [49] required clearing the 'CVM required' flag in the TTQ (because they were working in the UK with non-Tap-and-PIN cards), whereas Basin et al. [42] do not need to clear this bit (because they were working in Switzerland with a Tap-and-PIN card).

### 4.3 Offline Over-the-Limit Visa Transactions

**The Feature:** This is offline mode for Visa, specifically the transaction/card data Visa includes in the SDADs, such as to know what attacks can be caught by an offline reader.

**Our Experiments & Analysis:** We experimented with combining ATTACKs 1 and 2, and separately with a Tap-and-PIN capable plastic card, to try to get the *Square Terminal* to do an over-the-limit Tap-and-PIN transaction offline, however, after finishing the exchange of messages, the terminal always rejected the transaction, without bringing up the PIN interface. Therefore, we conclude that the *Square Terminal* has been programmed to always reject Tap-and-PIN in offline mode.

We tried combining our ATTACK 1 *VAS-replay-relay* with the over-the-limit, online attack of Galloway and Yunusov [49], in order to do unauthenticated high value Visa transactions offline. Galloway and Yunusov concluded that their TTQ-CTQ-flipping attack only works for Visa online, as offline Visa adds CTQ-date signed inside the SDAD. However, our investigation of the SDAD (detailed in Appendix D) shows that the SDAD actually includes a *copy* of the CTQ that is included in the Card Authentication Related Data (CARD) field. So, we experimented, in offline mode, with changing the CTQ and leaving the CARD field untouched, as follows.

1. *Reader-Offline Phase.* We set the *Square Terminal* in offline-mode and set it up for a high-value transaction. Then, we run the MiM-relay script between a plastic card and the *Square Terminal* updating it to unset the "CVM required" bit of TTQ (byte 2, bit 7) and set the "CDCVM performed" bit in CTQ (byte 2, bit 8). This caused the transaction to appear as if it was verified on a mobile device, and makes the offline-mode *Square Terminal* accept the high-value payments with plastic cards, without any sort of verification[26].

2. *Online-Checks Phase.* When the *Square Terminal* goes online, the payment is rejected and the transaction is recorded as declined on the Square server-side log.

**ATTACK 3:** For Visa, we can make the *Square Terminal* accept a plastic contactless card offline for over-the-limit transactions, by performing ATTACK 1 where its relay is enhanced to a MiM flipping bits in the "CVM required" flag in the TTQ (byte 2, bit 7) and "CDCVM performed" flag in the CTQ (byte 2, bit 8).

**Discussions.** We consider this a "free-lunch attack" because it allows an attacker to leave with high-value goods before the merchant connects to the Internet to see the rejection[27]. The Square system offers no way for the merchant to recover the lost money or follow up with the cardholder. Although the merchant might not be technically liable, reclaiming the money is difficult.

ATTACK 3 combines the Galloway and Yunusov over-the-limit attack [49] with our ATTACK 1. The key novelty is that, unlike Galloway and Yunusov's online attack, ATTACK 3 is offline, and so bypasses online checks. This attack is most likely due to an implementation error on *Square Terminal*s: not checking that the CTQ and its copy in the CARD field are equal, given the CARD field is signed in the SDAD.

### 4.4 Offline Over-the-Limit MC Transactions

Our analysis in Section 4.2 showed that, for over-the-limit transactions with Mastercard, Tap-and-PIN checks are handled online by the backend system. Unlike Visa, Mastercard's protections are enforced during this backend process, making it difficult for an offline terminal to perform these checks.

To test this, we used our *VAS-replay-relay* technique (ATTACK 1) to relay an over-the-limit transaction from the offline *Square Terminal* to a plastic Mastercard, which was accepted offline. This indicates the offline *Square Terminal* does not check the CVR in the IAD within the SDAD for Mastercard. The transaction is rejected when the terminal goes online, however, by this point, an attacker with a stolen card could have already left a shop with high value goods.

> **ATTACK 4:** For Mastercard, we can make the *Square Terminal* accept a plastic contactless card offline for over-the-limit transactions, by just doing ATTACK 1.

**Discussions.** To our knowledge, this is the only over-the-limit attack against Mastercard based on protocol specifications. While previous research found Mastercard's protocol more secure than Visa's, that work looked at online transactions. By examining the features discussed in the last section, we discovered that Mastercard's user authentication checks are performed by backend systems, only accessible to online terminals, therefore compromising Mastercard's protocol security for offline transactions.

## 5 Formal Modelling & Recommendations for Plastic-Cards Attacks

To ascertain how to best fix ATTACKS 1-4, we extend the most recent of these models [52], which uses the Tamarin prover for cryptographic protocols [53], together with our new findings. Tamarin lets us model protocol using multi-set rewriting rules and will attempt to automatically prove or find counter examples to lemma based on actions of the protocol. We direct the reader to the Tamarin manual [54] for more details and examples.

We enhanced the EMV models in [52] as follows:
1. Modelling offline readers. Reader sends the SDAD to the bank via two rules (not one). First, the reader checks the SDAD. To allow for clear offline mode, the transaction is sent to the bank in a second rule.
2. Distinguishing between Plastic and Mobile. We modelled VAS exchanges, as a way to differentiate between mobile phones and plastic cards, as seen on the *Square Terminal*.
3. Tap-and-PIN. As we found that Mastercard only controlled Tap-and-PIN transactions online, at the backend, we did not model this in our offline-EMV model. For Visa, we added a PIN number for each card, and Tap-and-PIN cards that set the CTQ to "Online PIN required" (Byte 1 Bit 8). On receiving this, the reader would ask for a PIN that is either provided on a secure channel from the card, or on a public channel from the attacker. A fresh name is used as the PIN number meaning that the attacker cannot guess it.

In line with our attacker model we do not consider comprised terminals, payment devices or banks. All of the model files are available on our repository and have detailed comments explaining how the models work.

**Correctness Properties.** In line with our attacker model, discussed in Section 3.2, we use three types of lemma to check the correctness of the protocols. First, "over the limit" attacks, that ask Tamarin to look for the existence of a protocol trace in which the terminal or the bank accept a high value transaction, from an unauthenticated payment device (a plastic card or a locked phone). Second, for the regulation checks, we write lemmas that ask Tamarin to test if a plastic card can be accepted offline and if non-Tap-and-PIN cards can be made to do Tap-and-PIN. Third, we check for "free lunch" attacks by asking Tamarin to find traces that are accepted by the offline reader but then rejected by the bank.

**Understanding Our Mastercard Over-the-Limit ATTACK 4.** We used Tamarin to create an over-the-limit trace accepted by the reader but rejected by the backend. The trace shows: 1. The faked VAS message mimics phone interaction. 2. The plastic-card's AIP lacks CDCVM support, so the terminal selects "No CVM performed." 3. The terminal checks and accepts the SDAD. 4. The backend rejects the transaction due to the absence of CDCVM. The formal model confirms that this attack would be caught by the bank, when the terminal goes online. Backend checks are not detailed in EMV specifications. We tested the model with/without various backend checks. The lack of IAD check at the backend matches the *Square Terminal*'s behaviour in our ATTACK 4.

### 5.1 Fixing Offline EMV for Plastic Cards

**5.1.1 Fixing Visa Offline.** For Visa, Byte 2, Bit 8 of the CTQ field sent from the payment device to the reader indicates if CDCVM authentication has been used, i.e., a phone is being used. When Visa is used in offline mode, this field is included in the signed SDAD and so its integrity can be checked by the reader (as per Section 2). So, making the offline reader check Byte 2, Bit 8 is set inside the CTQ, inside the SDAD, is a good way to stop these attacks, ensuring that only authenticated phones are used in offline mode. The formal model confirms this, so, it is our recommended solution for Visa.

**5.1.2. Fixing Mastercard Offline.** Mastercard does not have a publicly defined field equivalent to Visa's CTQ. So, we consider three possible fixes.

**a. Using the IAD.** The IAD field encodes the device type and indicates if the user was authenticated. For Mastercard, it is signed in the SDAD, allowing offline terminals to verify it. We tested whether verifying this information would stop ATTACK 4 by requiring that offline readers only accept transactions with signed IADs indicating CDCVM was performed. Tamarin confirms this stops the attack. However, since the IAD is proprietary data meant for the card issuer, it may not be easily or reliably used by third-party EMV manufacturers. I.e., our analysis shows that if Mastercard formalised the IAD and added it to the core EMV specification, many of these issues could be fixed easily.

**b. Using the "ICC verifies PIN" CVM Result.** The CVM List and CVM Result in the Mastercard protocol seem like logical places to enforce offline user authentication. We tested the effect of the offline *Square Terminal* always setting the CVM Result to "010002: Plaintext PIN verification performed by ICC (the device)", but Tamarin shows this does not stop the attack, as plastic cards allow this without user authentication. Testing the effect of stopping cards from responding to this CVM Result also failed when offline readers use "No CVM performed". However, combining both changes stops the attack, though it would require extensive updates to both cards and offline readers, taking years to implement.

**c. Checking CDCVM supported in the AIP.** The AIP field sent from the card to the terminal includes a flag (byte 1, bit 2), indicating if the device supports CDCVM, signaling it is a phone, as standard plastic cards cannot do CDCVM. We tested making the *Square Terminal* check this bit in the ICC/card certificate. Tamarin confirms this stops plastic-card ATTACKS 1-4, and we recommended this fix to Square. However, locked phones in transit mode also indicate CDCVM support without always authenticating the user. To ensure this fix works, we need to explore offline EMV transactions with locked phones, discussed in Section 6.

## 5.2 Responsible Disclosure for ATTACKS 1-4

We informed Visa of relevant issues in ATTACKS 1-3. About ATTACK 2, Visa said that UK regulations would soon change to permit such Tap-and-PIN payment in the UK too, so they would not issue a fix yet.

We informed Mastercard of our illicit over-the-limit transactions with plastic Mastercards (ATTACK 4), that this may be a wider problem, but that Square was engaged in their fix.

We also notified Square of ATTACKS 1-4. For ATTACK 1, we recommend they check the AIP in the Integrated Circuit Card (ICC) Certificate to detect mobile/CDCVM-capable devices. Square paused UK contactless offline payments while fixing ATTACK 1, and worked closely with us, to check fixes before wide deployment. From October 2023 to February 2024, ATTACKS 1, 3, 4 were resolved. Square implemented our recommendation, rewarding us with a USD5000 bounty. See Figure 5 below for a summary of disclosures and implemented fixes per attack.

# 6 Testing Features Centred on Mobile Devices

After Square fixed the issues described above, we shifted our research to mobile/CDCVM-capable devices. Using our MiMs as per Figure 2 and tests on various terminals online and offline, we identified **6 attacks** and **2 problems** with mobile payments. Figure 4 summarises our findings, showing how they fit into the EMV ecosystem.

## 6.1 Screen Unlock, In-Wallet CDCVM & Transit/Transport Mode Findings and Attacks

**The Features.** This is "mobile(-wallet) customer validation", which has seen recent changes[28] in 2023/2024. Mobile devices typically unlock via PIN, pattern, or fingerprint, with mobile wallets requiring additional authentication, like a fingerprint, FaceID, or PIN. When near an active EMV reader, the phone's NFC and mobile wallet activates. It is the user authentication inside the wallet (e.g., inside ApplePay) that should set the "CVM perfomed" bit in the sense of the EMV protocol. However, some vendors use the screen unlocking to mean "CVM performed". Transport mode, another feature, sees further alterations around this, with in-wallet authentication and screen unlock, sometimes, bypassed entirely.

### 6.1.1 GooglePay

In GooglePay, to make any-amount payments, screen unlocking (and not in-wallet authentication) is needed for non-transport transactions. Once the phone has been unlocked recently, no further in-wallet authentication is required.

**PROBLEM 5:** For screen-unlocked phones using GooglePay with Visa, Mastercard, we can make the EMV terminals (offline and online), as well as banks and payment networks accept over-the-limit transactions, relayed, without any other MiM actions. This makes unlocked phones with GooglePay more vulnerable to any-value relays than ApplePay (which unlocked does still need in-wallet authentication), for both Visa and Mastercard, and even more vulnerable than plastic cards (where a PIN-verification may be triggered if limits are attained).

**Our Experiments:** For transport retailers like train gates, GooglePay's transport mode allows payments with the default contactless card. If "no verification" is enabled in GoogleProfile settings, payments can be made without screen unlocking or in-wallet authentication, on a lit-up screen, potentially for any amount. We found that *Square Terminal*'s new

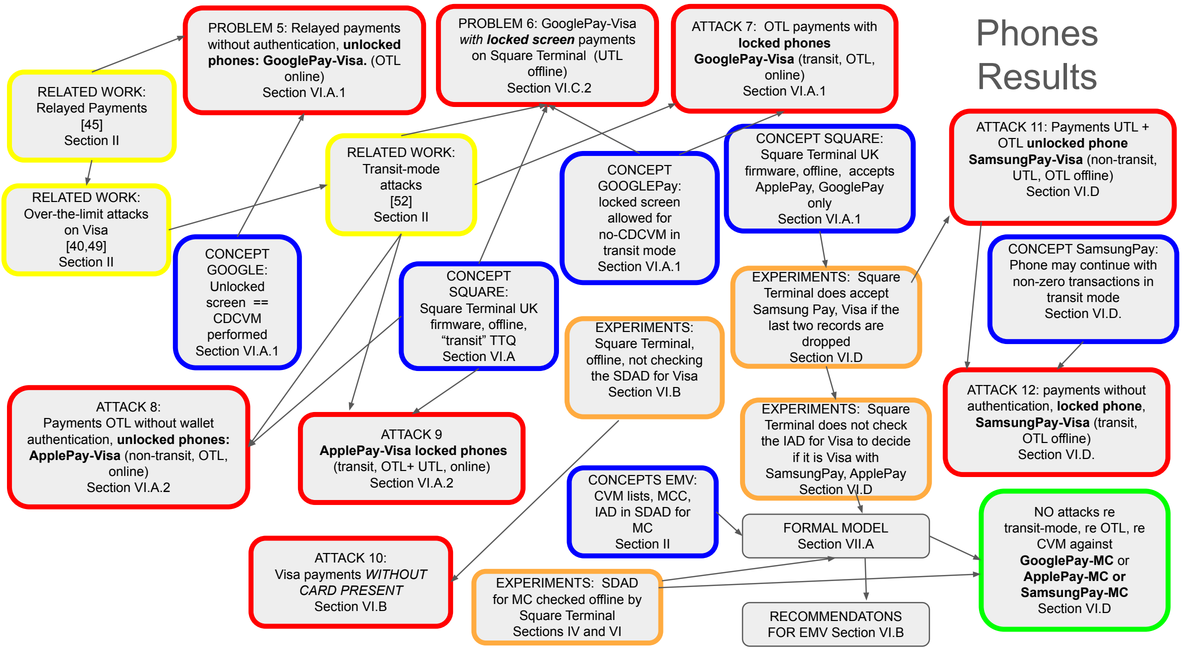
Phones Results
PROBLEM 5: Relayed payments without authentication, **unlocked phones: GooglePay-Visa.** (OTL online) Section VI.A.1
PROBLEM 6: GooglePay-Visa *with* ***locked screen*** payments on Square Terminal (UTL offline) Section VI.C.2
ATTACK 7: OTL payments with **locked phones GooglePay-Visa** (transit, OTL, online) Section VI.A.1
RELATED WORK: Relayed Payments [45] Section II
RELATED WORK: Over-the-limit attacks on Visa [40,49] Section II
RELATED WORK: Transit-mode attacks [52] Section II
CONCEPT GOOGLE: Unlocked screen == CDCVM performed Section VI.A.1
CONCEPT SQUARE: Square Terminal UK firmware, offline, “transit” TTQ Section VI.A
CONCEPT GOOGLEPay: locked screen allowed for no-CDCVM in transit mode Section VI.A.1
CONCEPT SQUARE: Square Terminal UK firmware, offline, accepts ApplePay, GooglePay only Section VI.A.1
ATTACK 11: Payments UTL + OTL **unlocked phone SamsungPay-Visa** (non-transit, UTL, OTL offline) Section VI.D
CONCEPT SamsungPay: Phone may continue with non-zero transactions in transit mode Section VI.D.
EXPERIMENTS: Square Terminal does accept Samsung Pay, Visa if the last two records are dropped Section VI.D
EXPERIMENTS: Square Terminal, offline, not checking the SDAD for Visa Section VI.B
ATTACK 8: Payments OTL without wallet authentication, **unlocked phones: ApplePay-Visa** (non-transit, OTL, online) Section VI.A.2
ATTACK 9 **ApplePay-Visa locked phones** (transit, OTL+ UTL, online) Section VI.A.2
EXPERIMENTS: Square Terminal does not check the IAD for Visa to decide if it is Visa with SamsungPay, ApplePay Section VI.D
ATTACK 12: payments without authentication, **locked phone, SamsungPay-Visa** (transit, OTL offline) Section VI.D.
CONCEPTS EMV: CVM lists, MCC, IAD in SDAD for MC Section II
FORMAL MODEL Section VII.A
NO attacks re transit-mode, re OTL, re CVM against **GooglePay-MC** or **ApplePay-MC or SamsungPay-MC** Section VI.D
ATTACK 10: Visa payments *WITHOUT CARD PRESENT* Section VI.B
EXPERIMENTS: SDAD for MC checked offline by Square Terminal Sections IV and VI
RECOMMENDATONS FOR EMV Section VI.B

Figure 4: Mind-map of the Main Results from Sections 6 and 7

firmware (shipped after patching ATTACK 1) is advertising, a TTQ with the “Offline Data Authentication for Online Authorizations" bit set (bit 1, byte 1).

**Results:** We found that a screen-locked GooglePay is using *only* this bit in the TTQ to identify a request for a transport transaction and activate its transport mode. This leads to:

**PROBLEM 6:** Screen-locked phones using GooglePay with Visa pay by the *Square Terminal* (offline) for under-the-limit transactions without any authentication, due to the *Square Terminal*’s “offline for online” TTQ, and banks and payment networks later accept it (online).

**Analysis.** The problem, video-ed here[29], lies in that GooglePay with Visa identifies transport in a weak way, just via the TTQ, without even looking at the MCC. It is unfortunate that the *Square Terminal* advertises an “offline for online” TTQ when it is completely offline. Worse, when the reader goes online, banks/payment networks do not stop this attack, e.g., via checking the MCC sent by Square.

**Our Experiments:** We lift this to an over-the-limit attack:

**ATTACK 7:** We can make screen-locked phones using GooglePay with Visa pay at a *Square Terminal* (offline), for over-the-limit transactions without any authentication, by unsetting “CVM required” flag in the TTQ (byte 2, bit 7), and setting “CDCVM performed” flag in the CTQ (byte 2, bit 8), à la ATTACK 3. Banks and payment networks later accept the transaction (online).

**Analysis:** Due to the *Square Terminal*’s TTQ, GooglePay thinks it is in transport mode. By looking at our traces (including traces[30] sniffed in the wild), *GooglePay sets CDVCM-done in the Visa CTQ when it runs in transport mode*; Also, the CVR in IAD, the CTQ in the SDAD in the trace of ATTACK 7[31] show CDCVM done. ATTACK 7 should be stopped: offline, by the *Square Terminal*(via the SDAD); and/or online, by the payment networks (via the IAD).

### 6.1.2 ApplePay

For ApplePay, a locked phone requires screen unlocking for EMV transactions. On top, if recently unlocked, a double-tap may suffice for full in-wallet authentication, depending on settings, country, and amount, as shown here[32].

**Our Experiments:** We bypass the double-tap or authentication using a MiM attack with TTQ/CTQ-flipping, similar to ATTACK 3, but without VAS-response replay (given we are on mobile phones):

**ATTACK 8:** For screen-unlocked phones using ApplePay with Visa, we can make the *Square Terminal* (offline), and banks and payment networks (online) accept over-the-limit transactions without ApplePay-wallet authorisation, by unsetting/setting the same TTQ/CTQ flags as in ATTACK 7.

**Analysis:** Like with ATTACK 7, the *Square Terminal* fails to check the CTQ in the SDAD and the payment networks do not check the CVR in the IAD, both of which will stop the attack Radu et al. [52] found that *Apple Enhanced Contactless Polling (ECP)*, US patent US11200557B2[33] is used to enter transit mode for Apple, when the screen is locked.

**Our Experiments:** We now attempt to lift ATTACK 8 to transit-mode attacks, using the methods in [52]:

**ATTACK 9:** For screen-locked phones using ApplePay with Visa, we can make the *Square Terminal* (offline), banks and payment networks (online) accept over-the-limit transactions without ApplePay-wallet authorisation; we replay Apple ECP bytes and change the *Square Terminal*'s MCC to a transport one, while –for over-the-limit– we also flip the CDCVM-relevant TTQ/CTQ flags, à la ATTACK 3.

**Analysis:** As per [52]'s attacks, there are lacks of backend checks of the TTQ against the MCC to see if the terminal is that of a transport merchant.

**Discussions:** It is the first time this is done *offline*, which was deemed improbable by [52], assuming the SDAD checks would be duly done.

## 6.2 No-Card-present Attack

**The Feature:** This is online readers which offer on-request offline capability, in compliance with different regulatory areas (e.g., the EU, the UK, the US).

**Our Experiments:** Via ATTACKS 7-9, we found that, in fact, the *Square Terminal*'s new firmware did not verify any data stored in the SDAD. So, we built a **card emulator that replays a sniffed Visa-card's certificates, and responds with fake EMV-format messages, including fake SDADs**.

**ATTACK 10:** We can make the *Square Terminal* accept offline any-value Visa payments, without any bank-issued card being present in the transaction. Online, these payments are declined due to missing valid cryptographic data.

**Analysis:** This amounts to us fabricating imaginary cards out of thin air, and using these fake cards to pay in shops. Again, the Square Terminal allows merchants to set limits on offline payments, up to a maximum of £25,000; the default limit is £100. We filmed our ATTACK 10 for £25,000[34].

**Discussions:** In the disclosure process, we found out that US Visa cards do not send SDADs for transactions with terminals that have TTQs with "offline for online" bit set. Visa US should adhere to the EMV specification and send SDADs, in those cases; UK/EU Visa cards do send SDADs in such cases. *Square Terminal* offline, should send a TTQ with the 'offline' bit set, as in its previous firmware, and –in all cases– check the SDADs and, if not sent, reject the transaction.

# 7 Mastercard: Mobile & Its CVM Results

**The Feature:** The feature analysed here is over-the-limit checks by Mastercard. The attacks on mobile phones above are all against Visa. To do similar over-the-limit attacks against Mastercard, we need to understand how the CVM options work for the Mastercard protocol, from the perspective of both the card/phone payment devices and of terminals.

**Our Experiments:** We ran experiments and noticed that the CVM List on most Mastercard payment devices[35] is "42031E031F03". This constitutes a list of three possible verification methods for the reader to choose from, with 4203 meaning "Enciphered PIN verified online, if terminal supports", 1E03 meaning "Signature, if terminal supports" and 1F03 meaning "No CVM required, if terminal supports". The EMV specification describes these options and states that the reader should pick one and return its choice, as the CVM Result, to the payment device.

Some phones, at the start of the CVM List, also have the option: 4201 denoting "Enciphered PIN verified online, if unattended cash". For plastic cards, the CVM List most commonly seen is "02031E031F03" [36], which means the same as the above "42031E031F03" CVM List seen on phones.

We experimented with plastic Mastercard cards, ApplePay and GooglePay, whilst locked and unlocked, and with the *Square Terminal*, both online and offline, as well as a terminal pretending to be transport/transit mode (i.e., sending ECP bytes before EMV transactions, to signal a transport operator).

**Results:** Table 1 shows the CVM result selected by the reader in each case. In some cases, the CVM Result is selected from the CVM List as expected, e.g., for over-the-limit, online, the reader attempts online PIN verification. However, the offline *Square Terminal* does not select a CVM from the list and instead uses the 3F00 code to warn the device that no CVM has been performed, the same behaviour has been observed in transit readers. Non-Tap-and-PIN cards could end the transaction , but –as per Section 4.2– they do not.

We found that *locked* ApplePay or GooglePay would end the transaction if given the CVM Result '010002' by the reader. This makes sense, since this CVM Result tells the device to authenticate the user, and a locked phone has not.

The CVM Result chosen by the offline *Square Terminal* for mobile phones also tells the device that it must perform CVM itself, i.e., "Plaintext PIN verification performed by ICC". This was not an option given in the CVM List.

**Analysis:** The experiments above confirm that, in line with the Mastercard specification, the reader selects the CVM Result "Plaintext PIN verification performed by ICC", whenever the AIP on the card indicates that on device cardholder verification is supported (Byte 1 Bit 2). Following the above, we experiment with *Square Terminal* and conclude:

**STATEMENT:** We are unable to do any CDCVM-based and/or transport-based attacks for mobile-wallets (ApplePay, GooglePay, SamsungPay) using Mastercard against the *Square Terminal* (offline), due to the way CVM options are treated in Mastercard's protocol and the *Square Terminal* checking offline the Mastercard SDAD.

**Discussions:** Mastercard's protocol handles CVM options better than Visa, authenticating all CVM-related fields inside the SDAD. And, the terminal needing to check all CVM-related fields in the SDAD, as it must check the SDAD to extract the AC, unlike with Visa.

| | Novelty/ Reliance on prior work | Attack Characterisation*: Breach of Regulation, Free-lunch, OTL (over-the-limit) | Root cause: Flawed technical specification, non-compliance with technical specification, bad implementation | Payment method | Payment network | Authorisation when the reader goes online | Victims | Scope: EMV-wide vs. Company | Impact** (via scope, victim, payment method) | Who can fix / Fixed or not |
|---|---|---|---|---|---|---|---|---|---|---|
| ATTACK 1 | **Novel**. No one looked at offline readers trying to distinguish plastic cards from other devices | **Regulation**: Square to block plastic offline in the UK | - **flawed specification**: Square wrongly creates a specification, whereby it it to use VAS to identify CDCVM-capable devices | Plastic | Visa<br>Mastercard | Rejected online | **Cardholder**: if their card was stolen | Square | low | **- Square**: to use AIP (indicative of CDCVM capability, authenticated in card cert)<br>**- disclosed** in October 2023;<br>- **fixed + bounty received** |
| ATTACK 2 | **Novel.** First to study '***non***-tap-n-pin' cards; prior work focused on 'tap-n-pin' cards (e.g., [49,52]); they missed this distinction entirely. | **Regulation**: UK Payment Services does not allow tap-n-pin cards in the UK | - **flawed specification**: Visa specifies to use a bit in the CTQ to encode (*non*-) TAP-n-PIN | Plastic | Visa | Accepted online | **Cardholder**: if their card was stolen | EMV-wide | high | **- EMV/Visa**: to securely signal non-tap-n-pin cards (e.g., via AIP);<br>**- attenuated by UK Payment Services** -- to phase out non-tap-n-pin cards |
| ATTACK 3 | **Novel - re the offline aspects.** As per ATTACK 1<br>**Incremental - re the over-the-limit aspects for Visa** [40,49,52]. But, [49] thought offline readers stop this, via SDAD checks; we refute this. | **Regulation:** Square to block plastic offline in the UK<br>**OTL**<br>**Freelunch** | - **flawed specification**: see ATTACK 1<br>- **bad implementation**: Square failed to check offline SDAD against illicit CTQ *(as admitted in disclosure)* | Plastic | Visa | Rejected online | **Merchant** (free-lunch attack)<br>+<br>**Cardholder**: if their card was stolen | Square | medium | **- Square** - re ATTACK 1 |
| ATTACK 4 | **Novel.** As per ATTACK 1<br>**Novel.** First to study over-the-limit authentication in Mastercard online+offline. | **OTL**<br>**Freelunch** | - **non-compliance with Mastercard specification:** Mastercard's over-the-limit customer authentication can only be checked online, not offline, so Mastercard plastic cards should not be used offline, for OTL; Square infringes that<br>- **flawed specification**: see ATTACK 1 | Plastic | Mastercard | Rejected online | **Merchant** (free-lunch attack**)**<br>+<br>**Cardholder**: if their card was stolen | All offline terminals (demonstrated on Square)<br>Mastercard | high | **- Square** - re ATTACK 1<br>**- EMV/Mastercard specifications**: to introduce offline checks for customer authentication (for plastic cards)<br>**- disclosed** in October 2023;<br>**- Fixed only for Square terminals** + Mastercard stated that they wanted Square to fix this and nothing else. |
| ATTACK 7 | **Novel**. Testing the new settings for transport-mode for Google.<br>**Incremental**. We simplify over-the-limit, transit attacks for Visa shown in [52], for the Square Terminal offline, as well as for some Google settings | **UTL/OTL** | **- flawed specification**: Google lacks sufficient EMV field checks to confirm transport terminals.<br>**- flawed specification**: Square's 2024 firmware wrongly sets the "offline for online" TTQ bit for *offline* terminals.<br>- **non-compliance with specification**: VISA's backends should verify the AC for transport-specific details. | GooglePay, locked, transit mode, no auth/CDCVM at all | Visa | Accepted online | **Cardholder**. phone has not been stolen, and it is locked. | GooglePay<br>Visa | medium | **- Google**: to check, e.g., MCC, etc., + ask for some form of user input or alert them in case of payment, for transport mode<br>**- Square**: to send TTQ with the "offline for online bit" unset, for offline terminals<br>**- Visa**: to check online the CDCVM bits inside the AC, as well as the MCCs.<br>**- disclosed + not fixed**. Square liaising with parties around this. |
| ATTACK 8 | **Incremental** - re the over-the-limit aspects for Visa in [40,49]; we change them from plastic to unlocked phones | **OTL** | - **bad specification:** payment allowed on an unlocked iPhone without further prompting to the user for authentication in the ApplePay;<br>- **non-compliance with specification**: VISA's backends should check the AC for all relevant details | ApplePay unlocked, no auth/CDCVM in-wallet | Visa | Accepted online | **Cardholder**: Their phone has not been stollen, all is needed is that it is unlocked. | ApplePay<br>Visa | weak | **- ApplePay:** to notify the user when a payment is made.<br>**- Visa:** to check online the CDCVM bits inside the AC<br>- **undisclosed to Apple and Visa**, since it falls under [49], which remains unfixed since 2022.<br>- **disclosed to Square**, as part of ATTACK 9 |
| ATTACK 9 | **Incremental** - re the over-the-limit, transit attacks for Visa shown in [52]. We simplify, for the Square Terminal offline, where we do not need to modify the TTQ. | **OTL** | - **non-compliance with specification**: VISA's backend should check the AC for all transport + authentication details | ApplePay, screen locked, transit mode, no auth/CDCVM at all | Visa | Accepted online | **Cardholder**: Their phone has not been stollen, all is needed is that it is unlocked. | ApplePay<br>Visa | medium | **- ApplePay** - to notify the user when a payment is made + to bind the ApplePolling (used with transport PoS) into the EMV stack<br>**- Visa** - should check online the CDCVM bits inside the AC, as well as the MCC<br>**- undisclosed to Apple, Visa**, since it falls under the same problems disclosed by [49] in 2022 which remain unfixed;<br>**- disclosed to Square** |
| ATTACK 10 | **Novel** - we created a card emulator that replay old EMV transactions | **OTL**<br>**Freelunch** | - **non-compliance with specification:** Square does not check Visa's SDADs, if their terminal is set in 'offline for online' mode *(as admitted in disclosure)* | Any | Visa | Rejected online | **Merchant** (free-lunch attack) | Square<br>Visa | high | **- Square**: to implement the EMV specification: (a) check the SDAD at all times; (b) use TTQs strictly for *offline* readers when applicable.<br>**- Visa**: US Visa cards to send SDADs for transactions with PoS with TTQs ('offline for online' bit set). UK/EU Visa cards do send SDADs in such cases.<br>**- disclosed** in May 2024;<br>**- not yet fixed**; Square is working with Visa US around this<br>**- bounty received** |
| ATTACKS 11/12 | **Novel** - we made SamsungPay be illicitly accepted offline by Square in the UK, by dropping some cards records | **Regulation:** Square to block SamsungPay in the UK, offline<br>**OTL**<br>**Freelunch** | - **bad implementation**: Square accepts Visa payments when not all the records are present *(as admitted in disclosure)* | Samsung Pay, unlocked / locked, no auth/CDCVM in wallet | Visa | Rejected online | **Merchant** (free-lunch attack)<br>+<br>**Cardholder** -- with phone locked/unlocked | Square<br>Visa | medium | **- Square**: to implement the EMV specification and check the SDAD at all times; the error here may be linked to the US-Visa cards problem above<br>**- disclosed** in May 2024<br>**- not yet fixed** |

* *Note: There are several dimensions that can be used as full characterisations of our attacks. This three-dimensional one was used also in the formal models; each attack falls in at least one of these three categories.*

** *Note: The full impact is shown via all columns, but that forms several preorders over our attacks and it is therefore hard to include in a simple way in the table.*

Figure 5: All Our Attacks' Dimensions at A Glance

### 7.1 SamsungPay & the *Square Terminal*

Our tests with various SamsungPay devices revealed the following behavior: in non-transport mode, even if the phone is unlocked, mobile-wallet customer validation is required for any transaction, which can be accessed from the locked screen. In transport mode, SamsungPay allows transactions through a locked screen without authentication, but only for zero-value transactions. In this mode, SamsungPay does not use proprietary protocols and, like GooglePay, relies on standard EMV fields such as TTQs and MCCs.

**Feature:** Square stipulates to accept only ApplePay and GooglePay (i.e., implicitly, not other mobile-wallets), offline, in the UK.

**Our Experiment:** We confirmed this via payment attempts: i.e., SamsungPay is rejected in the UK[37]. So, akin to ATTACKS 1-4 on illicit plastic, we asked ourselves if we can make *Square Terminal* accept SamsungPay against its specifications. And, we already knew from Section 7 that CDCVM- or transport-based attacks for Mastercard are impossible because the terminal checks Mastercard's SDAD. But, in Visa's case, we have:

**ATTACKS 11 & 12:** By unsetting the "CVM Required" flag in the TTQ (byte 2, bit 7), and setting the "CDCVM performed" (bit 8 in byte 2) in the CTQ, and dropping the last two (out of four) records of the payment device, we make the *Square Terminal* accept over-the-limit transactions offline from unlocked /locked phones using SamsungPay and Visa in normal/"Transport" mode, which is against the *Square Terminal* UK requirements. The *Square Terminal* accepts this offline, despite a warning on payment device from SamsungPay. The bank and payment networks reject this, when the reader goes online.

**Analysis:** In ATTACK 12, the payment device mistakenly enters transit mode due to the Square Terminal's TTQ, showing a warning (since it expects only zero values in transit/transport mode), but not stopping the transaction. In ATTACK 11, outside transit mode, the phone shows no error. Our method is to drop the last two records containing the "public-key remainder", preventing the *Square Terminal* from verifying the ICCc and SDAD; wrongly, *Square Terminal* accepts the transaction afterwards. During ATTACKS 11-12, we also found that the ***Square Terminal* does not use IADs to distinguish between ApplePay, GooglePay and SamsungPay.**

**Discussions:** This attack may be a wider issue for Square, and may relate to ATTACK 10 (Visa-US non-compliance), but we only used this technique for this Samsung-based attack.

## 8 Formal Modelling & Recommendations Centred on Mobile Devices

As in Section 5, we used formal modelling in Tamarin, and enhanced our models in Section 5 and the models from [52] to support our new findings on mobile-wallets in Section 6 , and check our suggested fixes with the new mobile features we discovered. We do not model the records-dropping ATTACKS 11-12 for SamsungPay, as this may be purely an implementation flaw; we are discussing with Square.

**Fixing Offline EMV for Mobile Wallets:** Our model leads to the following recommendations. Checking the Visa SDAD is obligatory. Checking CDCVM authentication done bit in the CTQ stops the SDAD from locked iPhones being accepted offline. However the attacks on GooglePay, based on Square's use of the "offline for online" TTQ flag and GooglePay interpreting this as a transit reader, remain. If Square switched to requesting a fully offline transaction in the TTQ, this issue would be fixed. Google could also consider revising detecting transport mode solely based on the TTQ bits. We do not model the SamsungPay attacks, since we do not fully understand their workings. However, offline/online terminals should never accept a transaction if they cannot check ICCc. And, SamsungPay in case of a warning on its payment device should not send a valid AC.

**Model Statistics:** On a 64GB, 16-core MacBookPro M3, our Visa model takes 29 minutes to check. We split the Mastercard model into separate files for ApplePay and GooglePay, each of which takes around 6 minutes to run.

**Responsible Disclosure for Mobile:** We disclosed our findings and recommendations to Square and Google, but not to Apple, as ATTACKS 8-9 are variations of known Apple Pay-Visa issues that Visa is reportedly fixing. Google has not responded yet. Square has been very responsive, working to implement the SDAD, CTQ, and ICC-based recommendations, and considering changes to their TTQ. Also, Square told us that business rules and deviations from specifications in the US might be causing these issues: some Visa-based devices do not respond to certain TTQs nor send the SDAD, highlighting inconsistencies in the EMV ecosystem. For a summary on this, please refer to Figure 5.

## 9 Conclusions

Examining a series of aspects (plastic cards, mobile, transit modes, offline terminals), we were able to shed light on a number of un-disclosed EMV-payment behaviours, especially those linked to add-on, mobile-centric features. Some of these lead to security shortcomings, of various types. We followed responsible-disclosure processes, as discussed, and a statement by Square is included in the following "Ethical Considerations" section. Overall, we were only able to find and fix these issues because we investigated and reverse engineered the proprietary features built by a range of companies. Individual companies may not have the ability to do this and, we argue, should not be expected to. Rather, EMV stakeholders should communicate their features with EMVCo and have them added to the public EMV specification.

## Ethical Considerations.

**Attacks, Reproducibility and Ethics.** In terms of EMV readers, we used:
– several *Square Terminal*s, under UK firmware. We used them for contactless EMV payments, both offline and online. Offline, in the UK, the *Square Terminal* is supposed to accept contactlessly only GooglePay and ApplePay[38].
– one online SumUp Air[39] configured for Romania, one online SumUp Air configured for the UK, one online SumUp Solo configured for the UK and one online Square Reader configured for the UK. As such, they deal with PIN-checking for contactless cards differently.

All plastic cards, mobile devices and readers/PoS we used are not comprised and registered to us (i.e., we do not victimise retailers or the general public). We use no personal data, or sensitive data, other than ours.

We present **10 attacks and 2 "problems"**. Some of them stem from similar problems (e.g., poor checking of a Visa field called "SDAD"), some are unrelated to this (e.g., on GooglePay's transport mode), some are on plastic cards, some on mobile, they have different victims, levels of complexity, etc. To appreciate these aspects earlier on, we refer the reader to Figure 5, to an overview table of all our attacks.

**Statement by Parties Involved in the Responsible Disclosure.** The only EMV party involved who gave us a statement to include is Square. This is as follows: *"At Square, protecting the security of our seller community is our highest priority. We recognize the important contributions that the security research community can make when it comes to finding potential bugs and we have offered a bug bounty program to the security research community since 2014.*

*We have collaborated with this research team since their discovery and responsible disclosure of their initial findings via our bug bounty program outlined in the following paper. This collaboration involved replicating the issue using the conditions detailed by the team (see Section 4) and working closely with the research team in the testing phase of the security update we designed for this condition.*

*We can confirm that a security update for the researchers' initial disclosure has since been deployed to all susceptible Square devices. And while we remain actively engaged with our network partners on the security of the payments ecosystem, we do not believe the additional findings represent an increased risk to sellers and buyers transacting with offline payments at this time.*

*We would like to thank these researchers for their vigilance in identifying and responsibly disclosing their findings, as well as their collaboration in testing the security update we developed for their initial disclosure".*

## Open-Science Considerations.

Our code (Sections 4, 6), formal models (Sections 5, 7), EMV traces (i.e., EMV transactions – Sections 4, 6) and videos (Sections 4, 6) are at `https://archive.softwareheritage.org/browse/origin/directory/?origin_url=https://gitlab.com/anonym_123/anon_payments/`.

We are committed to the principles of open science. Our research involve the generation of code, and related materials, and we disclose it all at this stage, for reproducibility purposes. We will continue to have it available, in free-to-use licences, after publication.

In fact, we will ensure that all our findings and methodologies are thoroughly documented and openly shared through the publication in various format, on open repositories as well. By doing so, we aim to contribute with new insights to the community and facilitate further research and discussion in the field.

## References

1. https://www.emvco.com/knowledge-hub/emv-chip-growth-continues-globally
2. https://www.emvco.com/specifications/
3. https://www.fortunebusinessinsights.com/point-of-sale-pos-market-106336
4. https://www.wearepay.uk/what-we-do/payment-systems/
5. https://squareup.com/us/en/point-of-sale/software
6. There are other banks involved, called acquiring banks or acquirers, but since we are not concerned with them, we can make this general
7. https://paymentcardtools.com/emv-tag-decoders/iad
8. https://www.jpmorgan.com/payments/client-resource-center/country-specific-payment-regulations
9. https://squareup.com/help/gb/en/article/7777-process-card-payments-with-offline-mode
10. https://www.regions.com/insights/small-business/operations/types-of-pos-fraud-how-to-prevent-them
11. https://proxmark.com/
12. https://www.sumup.com/en-gb/air-contactless-card-reader/
13. https://www.sumup.com/en-gb/solo-card-reader/
14. https://squareup.com/us/en/hardware/contactless-chip-reader
15. https://squareup.com/us/en/hardware/terminal
16. https://squareup.com/gb/en/about
17. https://support.apple.com/en-gb/guide/security/secbd55491ad/web
18. https://developers.google.com/wallet/smart-tap/introduction/overview
19. https://gist.github.com/gm3197

20. https://github.com/kormax/apple-vas

21. https://docs.springcard.com/apis/NET/AppleVAS/index.html

22. https://atlassian.idtechproducts.com/confluence/download/attachments/30479625/Apple%20VAS%20in%20ViVOpay%20Devices%20User%20Guide.pdf?api=v2

23. https://www.wearepay.uk/what-we-do/payment-systems/

24. For trace see https://archive.softwareheritage.org/browse/content/sha1_git:959fa33464a368b19448b828533fef13003fd7cd/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=attacks_plastic/traces/attack2-tapNPin/visa_otl_tapnpin_rejected.txt

25. See “Tap-and-PIN” attack traces here: https://archive.softwareheritage.org/browse/directory/9484e7ae7a6c345c0993943a6fbbfb049f1fbe79/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=attacks_plastic/traces/attack2-tapNPin.

26. For trace see https://archive.softwareheritage.org/browse/content/sha1_git:388445a4173bba858b616f550dce250cfbb61eb0/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=9484e7ae7a6c345c0993943a6fbbfb049f1fbe79/attacks_plastic/traces/attack3-Visa/visa_otl_plastic_CDCVM_trick.txt

27. See traces for the plastic-Visa offline attack, ATTACK 3, here: https://archive.softwareheritage.org/browse/revision/b118c5afb57162ae910a68a4ae1f580bdd112e73/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=attacks_plastic/traces/attack3-Visa&revision=b118c5afb57162ae910a68a4ae1f580bdd112e73 .

28. See, e.g., https://9to5google.com/2024/04/04/google-wallet-verification/

29. GooglePay entering transport-mode just based on TTQ; video evidence: https://archive.softwareheritage.org/browse/content/sha1_git:b9361f5dca554a4f67f9deda4fd15df68285f07c/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=9484e7ae7a6c345c0993943a6fbbfb049f1fbe79/attacks_mobile/traces_and_videos/helper_understanding/understanding_googleTransport/googlePay_enters_transport_mode_and_pays_without_auth_on_Square_due_to_TTQ.mp4 and https://archive.softwareheritage.org/browse/content/sha1_git:fdefa307169b78b1c0450ac86ec03fa632bd4ba3/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=9484e7ae7a6c345c0993943a6fbbfb049f1fbe79/attacks_mobile/traces_and_videos/helper_understanding/understanding_googleTransport/googlePay_does_not_enter_transport_with_onlineSumUp_as_TTQ_not_like_Square.mp4

30. See our sniffed GooglePay paying Transport for London here: https://archive.softwareheritage.org/browse/directory/9484e7ae7a6c345c0993943a6fbbfb049f1fbe79/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=attacks_mobile/traces_and_videos/helper_understanding/understanding_googleTransport .

31. See the ATTACK7 trace here: https://archive.softwareheritage.org/browse/content/sha1_git:b71527bcc5dccf20c9fd224fde1568726e57c7e9/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=9484e7ae7a6c345c0993943a6fbbfb049f1fbe79/attacks_mobile/traces_and_videos/attack7-googleLocked/offline_OTL_transit_attack7.txt .

32. ApplePay requires actions and authentication in the wallet to pay; see here https://archive.softwareheritage.org/browse/content/sha1_git:f86b957ad5f589722de8866054f0cccea9596084/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=9484e7ae7a6c345c0993943a6fbbfb049f1fbe79/attacks_mobile/traces_and_videos/helper_understanding/understanding_googleTransport/applePay_does_not_enter_transport_mode_an_pays_with_auth_on_Square_despite_to_TTQ.mp4.

33. See Apple’s ECP here: https://patents.google.com/patent/US11200557B2/en and https://github.com/kormax/apple-enhanced-contactless-polling

34. Paying £25,000 with an emulated imaginary card by an offline *Square Terminal*: https://archive.softwareheritage.org/browse/content/sha1_git:bded48b2aedcb8d12f9a6f82aa55b75607c0ce23/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=9484e7ae7a6c345c0993943a6fbbfb049f1fbe79/attacks_mobile/traces_and_videos/attack10-noCard/paying25k_withNoCard.mp4.

35. For trace, see https://archive.softwareheritage.org/browse/content/sha1_git:d9c476682da86609897dc0496e78d7e61269fc0d/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=9484e7ae7a6c345c0993943a6fbbfb049f1fbe79/attacks_plastic/traces/helper_understanding/traces_CVMResults/MCapplephonelockedonlineUTL.txt

36. For trace see https://archive.softwareheritage.org/browse/content/sha1_git:722096b0bef02298a1272396da95a6e6aef72090/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=attacks_plastic/traces/helper_understanding/traces_CVMResults/MCplasticonlineproxsniff.txt

37. https://archive.softwareheritage.org/browse/content/sha1_git:224b744281a6ce64e9df20b7d09a9cdd78452801/?origin_url=https://gitlab.com/anonym_123/anon_payments/&path=9484e7ae7a6c345c0993943a6fbbfb049f1fbe79/attacks_mobile/traces_and_videos/helper_understanding/understanding_samsungPay_onSquare/noSamsungPay-accepted-normally-on-square.mp4

38. https://squareup.com/help/gb/en/article/7777-process-card-payments-with-offline-mode

39. https://www.sumup.com/en-gb/air-contactless-card-reader/

[40] 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*, 2021.

[41] 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, Anaheim, CA, August 2023. USENIX Association.

[42] David A. Basin, Ralf Sasse, and Jorge Toro-Pozo. The EMV standard: Break, fix, verify. In *Security and Privacy (SP)*, 2021.

[43] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. ProVerif 2.00: Automatic cryptographic protocol verifier, user manual and tutorial, 2018.

[44] Ioana Boureanu, Tom Chothia, Alexandre Debant, and Stéphanie Delaune. Security Analysis and Implementation of Relay-Resistant Contactless Payments. In *Computer and Communications Security (CCS)*, 2020.

[45] 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 (FC)*, LNCS, 2015.

[46] J. de Ruiter and E. Poll. Formal analysis of the emv protocol suite. In *Theory of Security and Applications - Joint Workshop, TOSCA*, 2011.

[47] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactionson Information Theory 29*, 29(2), 1983.

[48] Martin Emms, Budi Arief, Nicholas Little, and Aad van Moorsel. Risks of offline verify pin on contactless cards. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security*, pages 313–321, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[49] L.-A. Galloway and T. Yunusov. First contact: New vulnerabilities in contactless payments. In *Black Hat Europe*, 2019.

[50] ISO. 7816-4: 2020 – Identification cards – Integrated circuit cards Part 4: Organization, security and commands for interchange. Standard, International Organization for Standardization, 2020.

[51] S. Meier, B. Schmidt, C. Cremers, and D. A. Basin. The tamarin prover for the symbolic analysis of security protocols. In *Computer Aided Verification - 25th International Conference, CAV*, 2013.

[52] 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, 2022.

[53] The Tamarin Team. *Tamarin-Prover Manual*. Security Protocol Analysis in the Symbolic Model. 9 2021.

[54] The Tamarin Team. *Tamarin prover manual*. The Tamarin Team, 2024.

[55] Aleksei Stennikov Timur Yunusov, Artem Ivachev. New vulnerabilities in public transport schemes for apple pay, samsung pay, gpay. *White Paper*, 2021.

[56] Visa. Visa merchant data standards manual, 2019.

[57] Timur Yunusov. Hand in your pocket without you noticing: Current state of mobile wallet security. *Black Hat Europe*, 2021.

## A Online vs Offline EMV, At a Glance

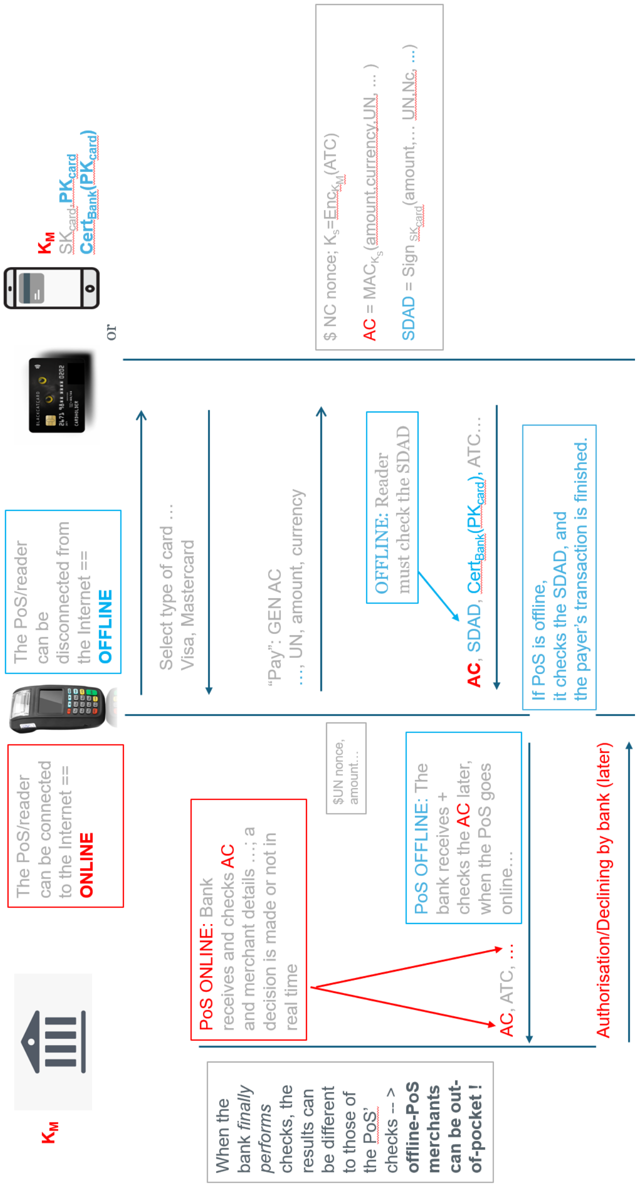
$K_M$
$SK_{card}$, $PK_{card}$
$Cert_{Bank}(PK_{card})$
or
$ NC nonce; $K_s$=$Enc_{K_M}$(ATC)
AC = $MAC_{K_s}$(amount,currency,UN, … )
SDAD = Sign $_{SK_{card}}$(amount,… UN,Nc, …)
The PoS/reader can be disconnected from the Internet == OFFLINE
Select type of card …
Visa, Mastercard
"Pay": GEN AC
…, UN, amount, currency
OFFLINE: Reader must check the SDAD
AC, SDAD, $Cert_{Bank}(PK_{card})$, ATC…
If PoS is offline, it checks the SDAD, and the payer's transaction is finished.
The PoS/reader can be connected to the Internet == ONLINE
$UN nonce, amount…
PoS ONLINE: Bank receives and checks AC and merchant details …; a decision is made or not in real time
PoS OFFLINE: The bank receives + checks the AC later, when the PoS goes online…
AC, ATC, …
Authorisation/Declining by bank (later)
$K_M$
When the bank *finally performs* checks, the results can be different to those of the PoS' checks --> **offline-PoS merchants can be out-of-pocket !**

Figure 6: Online & Offline EMV in Visa & Mastercard

## B More on Our Methodology

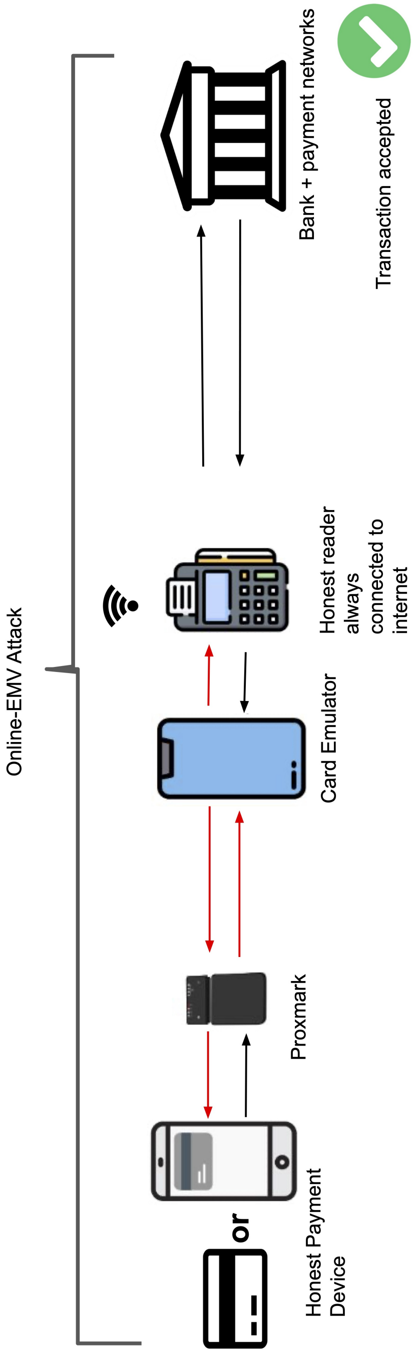

Figure 7: Our Methodology for Online-EMV Attacks (arrows form a relay-based MiM attack; red arrows – active MiMs)

## C The VAS Protocol – Details

The first step is for the reader to select the VAS app, by sending the "SELECT OSE.VAS.01" message to the phone. The phone replies with a TLV message, which uses different tag meanings to EMV. The tags we have seen are: 50: application name e.g., 'ApplePay', 9F21: version e.g., 0100, 9F23 capabilities e.g., VAS supported, and 9F24 to provide a nonce.

The reader then issues a GET DATA command. This command includes the pass type ID, a nonce, terminal capabilities and optionally a URL. The phone then generates an ephemeral ECDH key and uses the ANSI X9.63 SHA 256 key derivation function to combine this with the long term key and nonces to make a session key. The pass data is then encrypted with AES128 GCM and sent to the reader with the ephemeral key.

## D Decoding the SDAD

To the best of our knowledge, past papers have refered to the EMV specification for this, however the specification documents are quite vague, so some guess work has been used. In this section we decode the certificates and the SDAD, so we can clearly demonstrate what exactly is authenticated.

The card holds two certificates: 1) the Issuer's; 2). the ICC's (card's). During the transaction it may also generate the signed SDAD. Each of these are a single block of RSA. The reader holds the public key certificates of the issuers, which are publicly available, e.g., https://www.eftlab.com/knowledge-base/list-of-ca-public-keys and the card indicates which key was used to sign the Issuer Certificate in the Key Index field sent by the card. The reader applies RSA to decrypt this Issuer Certificate, e.g.,

```
Issuer Certificate from Visa card:
  Cert Header: 6a
  Certificate Format: 02
  Leftmost PAN digits: 483205ff
  Cert Expiration Date: 1223
  Cert Serial Number: 037691
  Hash Algorithm Indicator: 01
  Issuer Public Key Algorithm Indicator: 01
  Issuer Public Key Length: b0
  Issuer Public Key Exponent Length: 01
  Leftmost Bytes of the Issuer Public Key:
  b7122d435423228412c77896488bee7056f81109
  ... 1f7e89cfabf64928f31
  Hash: 0d4bfa9a3c98a03dc57c1e4c6af34ed71d
  1ee98d
  Trailing byte: bc
Hashed data is the ICC cert data expect
header, hash and trailing byte + ICC
Public Key Expo
```

We found the hashed data by trial and error. Mastercard contain more data in the hash, i.e., the ICC cert data, ICC Public Key Exponent, all of the second record and the AIP.

As the signature can be no longer than the key, the Issuer Certificate only contains the leftmost bytes of the key used for the ICC Certificate. The right most bytes are sent in the clear in READ RECORDS. Putting these bytes together makes it possible to decrypt the ICC Certificate as well, which is similar to the one above.

The SDAD is shorter than the ICC or Issuer Certificate so the entire SDAD key fits inside the ICC Certificate. Using this we can decrypt the SDAD:

```
SDAD:
  Cert Header: 6a
  Certificate Format: 95
  Hash Algorithm Indicator: 01
  ICC Dynamic Data Length (hex): 03
  ICC Dynamic Data (ATC): 0200b4
  Padding: bbbbbbbbbbbb...bbbbbbbbb
  Hash: 733f8f64b5a2e9f0b9147282e398050121
  adb47a
  Trailing byte: bc
Hashed data is the UN, Amount, Currency and
CARD data
```

We found that Visa included less data in the SDAD than some previous authors have claimed. However, the card data, in the hash, does contain the CTQ, so the reader can check the integrity of this field. The Mastercard SDAD contains more information, including the card records and the IAD.

## E Decoding CVM Results

| | Offline UTL | Offline OTL | Online UTL | Online OTL |
|---|---|---|---|---|
| Plastic Card (offline) | 3F0001 | 3F0001 | 1F0302 | 020300 |
| Apple/Google Pay (locked or unlocked) | 010002 | 010002 | 3F0002 | 010002 |

(a) 3F0001: No CVM performed; No CVM performed; Failed (or no CVM Condition Code satisfied); (b) 1F0302: No CVM required; If terminal supports the CVM, Successful; (c) 3F0002: No CVM performed; No CVM performed; Successful; (d) 020300: Enciphered PIN verified online; If terminal supports the CVM; Unknown; (e) 010002: Plaintext PIN verification performed by ICC; No CVM performed; Successful

Table 1: CVM Results with the *Square Terminal*

## F Diagram of Mastercard's Protocol with Relevant Details Highlighted

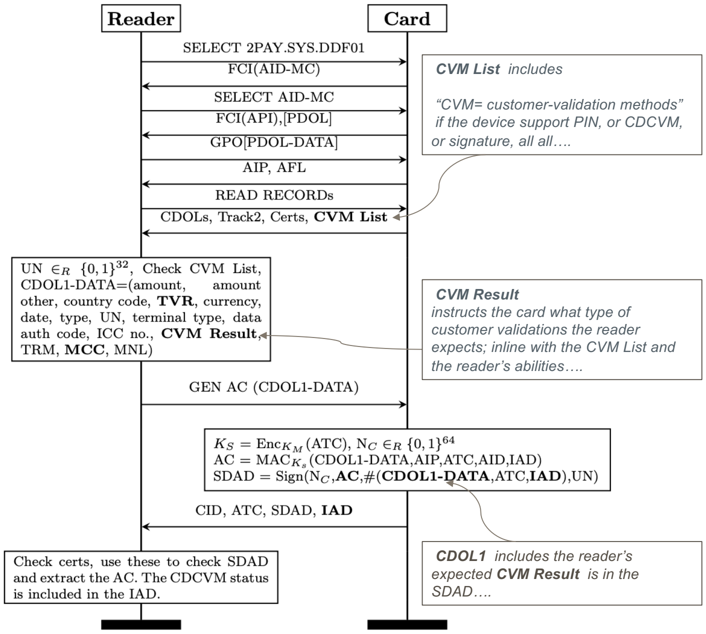
Reader
Card
SELECT 2PAY.SYS.DDF01
FCI(AID-MC)
SELECT AID-MC
FCI(API),[PDOL]
GPO[PDOL-DATA]
AIP, AFL
READ RECORDs
CDOLs, Track2, Certs, **CVM List**
**CVM List** includes
"CVM= customer-validation methods" if the device support PIN, or CDCVM, or signature, all all….
UN $\in_R$ $\{0,1\}^{32}$, Check CVM List, CDOL1-DATA=(amount, amount other, country code, **TVR**, currency, date, type, UN, terminal type, data auth code, ICC no., **CVM Result**, TRM, **MCC**, MNL)
**CVM Result** instructs the card what type of customer validations the reader expects; inline with the CVM List and the reader's abilities….
GEN AC (CDOL1-DATA)
$K_S$ = Enc$_{K_M}$(ATC), N$_C$ $\in_R$ $\{0,1\}^{64}$
AC = MAC$_{K_s}$(CDOL1-DATA,AIP,ATC,AID,IAD)
SDAD = Sign(N$_C$,**AC**,#(**CDOL1-DATA**,ATC,**IAD**),UN)
CID, ATC, SDAD, **IAD**
Check certs, use these to check SDAD and extract the AC. The CDCVM status is included in the IAD.
**CDOL1** includes the reader's expected **CVM Result** is in the SDAD….

Figure 8: Mastercard's PayPassProtocol (Most-relevant EMV fields in bold face and explained).

# G Relevant EMV Fields and Bits

| | | |
|---|---|---|
| AC | Application Cryptogram | A cryptogram generated by the card and used by the Issuer to confirm the legitimacy of the transaction. |
| AFL | Application File Locator | A list of application data records stored on the card. This is used to indicate to the terminal what data should be used when processing the transaction. |
| AID | Application Identifier | An Application Identifier uniquely labels an Europay, Mastercard, and Visa (EMV) application. A card reports to a reader the AIDs programmed into it, and the reader will select a supported one to process a transaction. |
| AIP | Application Interchange Profile | This indicates to the terminal what processing should be done for the transaction. |
| ATC | Application Transaction Counter | A counter stored on the card and incremented for each transaction. The Issuer monitors this value which should always increase. |
| CDCVM | Consumer Device CVM | The authentication method used on the consumer's own device for cardholder verification. |
| CDOL | Card Risk Management Data Object List | This list specifies the data to be used when generating the AC. |
| CID | Cryptogram Information Data | This data is returned by the card as part of the response to the Generate AC command. It is used to indicate the type of cryptogram and the actions to be taken by the terminal. |
| CTQ | Card Transaction Qualifier | This is set on the card when it is issued and is used to control what actions the terminal should take during a transaction. |
| CV | Cardholder Verification | Verifying that it is the legitimate cardholder who is making the transaction. |
| CVMs | Cardholder Verification Methods | These are used to verify that the right cardholder is present during the payment. |
| CVR | Card Verification Results | Data returned to the Issuer as part of the IAD. |
| FCI | File Control Information | This is a template that gives information about the data fields that follow. The FCI is not specific to EMV, it is part of the Level 2 specification (ISO/IEC 7816-4 [50]). |
| GEN AC | Generate AC | The instruction to generate the application cryptogram. |
| GPO | Get Processing Options | A command sent to the card with the requested PDOL data to retrieve transaction specific application data (AFL and IAD). |
| IAC | Issuer Action Code | This specifies what action the Issuer wants to be taken based on the TVR. Possible actions are: default, denial and online. |
| IAD | Issuer Application Data | Proprietary application data to be used in the transaction. |
| ICC | Issuer Country Code | Indicates the country of the card issuer. |
| MCC | Merchant Category Code | An ISO 18245 code used to classify the business type. |
| PDOL | Processing options Data Object List | A list of data sent to the card for it to use when processing the transaction. |
| SDAD | Signed Dynamic Application Data | A digital signature on the data used for DDA, or SDA. |
| Track 2 | User's account information on the card | The location of the user's data on the card. |
| TRID | Token Requestor ID | ID which uniquely identifies the pairing of Token Requester with the Token Service Provider. |
| TRM | Terminal Risk Management | The processes carried out on the terminal to protect from fraud. |
| TTQ | Terminal Transaction Qualifier | Data fixed on the terminal detailing its abilities and requirements. |
| TVR | Terminal Verification Results | This 5 byte result contains flags that show the result of the different processing functions that have been carried out as part of the transaction. |
| UN | Unpredictable Number | A nonce used as part of a transaction. Both the terminal and the card use nonces. |

| **Relevant EMV field** | **Relevant EMV flag in this field** | **Position in field** |
|---|---|---|
| AIP | On device cardholder verification supported | Byte 1, Bit 2 |
| CTQ | Online PIN Required<br>CDCVM performed | Byte 1, Bit 8<br>Byte 2, Bit 8 |
| TTQ | Offline Data Authentication for Online Authorizations supported<br>CVM required | Byte 1, Bit 1<br>Byte 2, Bit 7 |

Table 3: Relevant EMV fields, flags and their position.