Practical EMV Relay Protection

Andreea-Ina Radu*, Tom Chothia*, Christopher J.P. Newton†, Ioana Boureanu† and Liqun Chen†
*University of Birmingham, UK †University of Surrey, UK

***Abstract*—Relay attackers can forward messages between a contactless EMV bank card and a shop reader, making it possible to wirelessly pickpocket money. To protect against this, Apple Pay requires a user's fingerprint or Face ID to authorise payments, while Mastercard and Visa have proposed protocols to stop such relay attacks. We investigate transport payment modes and find that we can build on relaying to bypass the Apple Pay lock screen, and illicitly pay from a locked iPhone to any EMV reader, for any amount, without user authorisation. We show that Visa's proposed relay-countermeasure can be bypassed using rooted smart phones. We analyse Mastercard's relay protection, and show that its timing bounds could be more reliably imposed at the ISO 14443 protocol level, rather than at the EMV protocol level. With these insights, we propose a new relay-resistance protocol (`L1RP`) for EMV. We use the Tamarin prover to model mobile-phone payments with and without user authentication, and in different payment modes. We formally verify solutions to our attack suggested by Apple and Visa, and used by Samsung, and we verify that our proposed protocol provides protection from relay attacks.**

I. Introduction

Contactless Europay, Mastercard, and Visa (EMV) payments are a fast and easy way to make payments and are increasingly becoming a standard way to pay. However, if payments can be made with no user input, this increases the attack surface for adversaries and especially for *relay attackers*, who can ferry messages between cards and readers without the owner's knowledge, enabling fraudulent payments. Payments via smart-phone apps generally have to be confirmed by a user via a fingerprint, PIN code, or Face ID. This makes relay attacks less of a threat.

Apple Pay uses the EMV standard, however, for usability, iOS 12.3 (May 2019) introduced the "Express Transit/Travel" feature that allows Apple Pay to be used at a transport-ticketing barrier station without unlocking the phone. In October 2019, Samsung introduced the same feature. We refer to this feature as "Transport mode". We found that a non-standard sequence of bytes is broadcast by Transport for London (TfL) ticket-gate readers, and that these "magic bytes" bypass the Apple Pay lock screen. Apple Pay then checks that its other requirements are met (different for Visa and Mastercard) and if so it allows a transaction to be performed with no user interaction.

For Apple Pay Visa we alter, replay and relay both ISO 14443 Level 1[1] messages, as well as EMV protocol Level 3 messages; with this, we are able to make a fraudulent payment from a *locked* iPhone to any EMV shop reader (with non-transit merchant codes), for any amount; we tested up to £1000. For Mastercard, we found that relays from locked phones were only possible to readers with a transit merchant code. We formally model these protocols and verify the results, using the Tamarin prover; we extend the state-of-the-art EMV models from [2] to support mobile apps in different modes.

We disclosed this attack to both Apple and Visa, and discussed it with their security teams. Apple suggested that the best solution was for Visa to implement additional fraud detection checks, explicitly checking Issuer Application Data (IAD) and the Merchant Category Code (MCC). Meanwhile, Visa observed that the issue only applied to Apple (i.e., not Samsung Pay), so suggested that a fix should be made to Apple Pay. We verify Apple's and Visa's possible solutions in Tamarin and show that either would limit the impact of relaying. At the time of writing neither side has implemented a fix, so the Apple Pay Visa vulnerability remains live.

We found that Samsung Pay did not use "magic bytes", instead it was always possible to perform an EMV transaction with a locked Samsung phone. However, we found that locked Samsung-Pay would only allow a zero-value payment, requiring the transport providers (currently only TfL) to have an arrangement with their banks to charge for tickets based on these zero value transactions. This makes it impossible to relay Samsung Pay to shop readers to buy goods, but it is still possible to relay Samsung Pay to other transport readers.

It seems unlikely that transport modes will be removed from phones so, as relaying attacks are still possible, there is a need for general EMV relay-countermeasures. Visa have proposed a relay-countermeasure [3]; their protocol binds the ISO 14443 Level 1 data to the Level 3 protocol on the presumption that (relay) attackers cannot easily tamper with Level 1 data. This protocol has yet to be fully specified and implemented. Mastercard specifications include a Relay Resistance Protocol (RRP) [1]. This has been in the specifications since 2016, but as far as we are aware it is not yet implemented in commercial readers, or on customers' cards. RRP operates at the EMV application layer. In this protocol, the reader times a nonce exchange with the card. If the time taken to communicate with the card is within the bounds provided by the card, it is likely that the card is close to the reader and no relay attack is taking place. If the time taken is outside these bounds then the nonce exchange is repeated. If three nonce exchanges fail, then payment is rejected as a possible relay.

We show that the previously proposed relay-protection protocol from Visa can be defeated with standard hardware: the Level 1 data they bind into Level 3 can be forged. The Level 1

[1]In EMV terminology [1], Level 1 is the ISO 14443 protocol, Level 2 is the exchange of bytes encoded as Application Protocol Data Units (APDUs), and Level 3 corresponds to the EMV application protocol. We cover the ISO 14443 protocol in Appendix A and the EMV protocol in Section II-A. We do not manipulate the APDUs and so they are not detailed further.

data used comes only from the card, without involvement from, or timing by, the reader. This leaves the protocol open to Man-in-the-Middle (MitM) attacks. When contacted, Visa stated that their proposal provides protection against off-the-shelf smart phone relays. However, we show that a replay/relay is possible with standard phones, if one of them is rooted.

To test the Mastercard RRP, we measured the timings of responses coming from a test RRP-capable card, provided by ICC Solutions, as well as a range of other payment cards. We find that the distance of the card from the reader has a noticeable effect on the response times for the card's Level 3 messages, including the timed nonce-exchange on the RRP card. The Level 1 messages, which require less processing, show much shorter and more consistent response times. As users will commonly place payment cards at a range of distances and angles from the reader, it may be difficult for the reader to tell the difference between a card at the optimum position being relayed, and a legitimate card in the worst position. In the case of our test RRP-capable card, we show that this difference is enough to make a relay possible. While other card implementations may have regular enough timing to make a relay more difficult, relays are also becoming faster, even with cheap off-the-shelf hardware [4]. Requiring the user to place their card in a fixed position would mitigate this problem, but it would not be a usable solution.

We propose the `L1RP` protocol that combines elements from both Visa's and Mastercard's relay protections. We leverage the timed nonce-exchange as proposed by Mastercard, yet we move it to the ISO 14443 Level 1 part of the EMV protocol, which –as we show– gives stable round trip time (RTT) measurements. From Visa, we take the idea of tying together data from Level 1 with the EMV application authentication. By having a nonce-exchange and not just a one-directional message as in Visa's case, we solve the replay/relay issue with Visa's proposal. Our `L1RP` protocol satisfies the requirements of the ISO 14443 specifications and can be made backwards compatible, allowing both cards and readers to complete a transaction with a legacy reader or card (i.e., one without our relay protection).

We formally verify our `L1RP` protocol, and prove it secure using Tamarin. For relay-security, we rely on a previously published method in [5]. Concretely, we show that no downgrade attacks (to the EMV protocol without relay protection) are possible, and that our design provides the relay protection, as described in the threat model section below. We also implement our protocol's Level 1 nonce-exchange using Proxmarks [6], providing some evidence that our protocol is practical.

Novelty & Positioning: There have been a range of attacks against contactless EMV (e.g., [7], [2], [8]). Most of these attacks make use of a relay to alter messages going between a card and a reader, i.e., the relay aspect of our work is not new. However, we note that (1) None, of these past attacks work against EMV payments from *mobile devices* that require strong user-authentication, (2) None of these past attacks would work if good relay-protection protocols were in place, and we are the first to show that neither Visa, nor Mastercard's current relay-protection solutions are reliable.

Our work on mobile payments and transit modes adds important new insights to the field. For instance, Basin *et. al.* [2] present an attack that bypasses the contactless-transaction limit for Visa plastic cards by making it look like the card has used user authentication (CDCVM) when the card is not even capable of it, and they suggest changes to Visa's protocol to fix this. However, we show that the CDCVM status of a capable device is recorded in the IAD field in Visa's protocol, and this field may be authenticated by the bank, so all Visa need to do to stop this attack is to check this existing field in their protocol. So, we delay discussion of related work until the end of the paper, after we have presented our new findings.

The contributions of this paper are:

1) Explaining how Transit/Transport mode and the Issuer Application Data work and are used in EMV.
2) Showing how to bypass the Apple Pay lock screen take any amount of money from a Visa on an iPhone.
3) Showing that Visa's Level 1-relay-protected protocol is insecure against an attacker with rooted phones
4) Showing that EMV distance bounding can be done more reliability at Level 1 than Level 3.
5) Proposing a Level 1 distance bounding protocol for EMV.

II. Background

A. Overview of EMV

The EMV standard includes many different protocols, with many variations. We present the versions of Mastercard's PayPass and Visa's PayWave that we observed in mobile phone transaction traces. All of the annotated traces we collected can be found in [9]. We summarise the most important acronyms used by EMV in Appendix D.

1) Mastercard's Protocol: The version of PayPass we observed is shown in Fig 1. It runs after the Level 1 ISO 14443-3 anti-collision protocol; the relevant parts of the ISO 14443 protocol are described in Appendix A.

The card and the bank share a symmetric key K_M, and the card has a certificate chain for a public key, which the reader can verify. The first two messages exchanged select the payment application (i.e., Mastercard). Next, the reader sends a Get Processing Options (GPO) message with terminal-specific information, and the payment device answers with a list of the records available on the card (the Application File Locator (AFL)) and a list of the card's capabilities (the Application Interchange Profile (AIP)). The AIP indicates whether the device is capable of user authentication, but does not indicate whether user authentication has actually been used.

The reader will then request all of the records listed in the AFL, which includes "Track 2" (the user's account information) and the Card Risk Management Data Object List 1 (CDOL1), which lists all of the information the card needs to complete the transaction. The information requested may vary between cards; however, for mobile devices using Mastercard this always includes the amount of the transaction, a unique number/nonce from the reader (Unpredictable Number (UN))

and the MCC, which identifies the business associated with the reader (e.g., 5732:electronics stores or 4111:local transport).

The proof of payment that the reader requires from the card is a MAC on the transaction data, referred to as an Application Cryptogram (AC). The reader now requests this AC with a GEN AC command, which sends all of the data the card requested in the CDOL.

The card will then generate a session key, K_S, based on the key it shares with the bank, K_M, and its Application Transaction Counter (ATC), which represents the total number of times the card has been used. The card generates the AC as a MAC of the CDOL1 data, ATC, and AIP, keyed with K_S.

The reader cannot check this MAC as the key is known only to the bank and card, so the card signs data for the reader to check, the Signed Dynamic Application Data (SDAD), which includes the CDOL1 data, the AC, the AIP (if present in the Static Data Authentication Tag List), any records marked to be included in data authentication and the UN.

The SDAD and the AC are sent by the card to the reader along with the ATC, needed by the bank to calculate the MAC key, a CID (which indicates the type of the AC) and the IAD (which we discuss below). The reader checks the SDAD signature and the data in the SDAD and, if this is correct, it sends the AC, the AIP, CDOL1 data, ATC and IAD to the bank/payment network, which will verify the AC. If it is correct the bank will authorise the payment. The MCC is also sent securely to the bank, by the terminal, as part of this "authorisation request message".

We note that there are many variations of this protocol, e.g., the specification includes a card nonce, N_c, which is included in the SDAD, however we did not see this in any of our runs. The protocol presented here uses Combined DDA with application cyptogram (CDA) mode, as specified in the AIP; the specification allows for an "online" mode without a SDAD, although we have not seen this for any of our tested cards and readers, which are online and still use CDA.

Below, we present the Visa protocol, which is similar to Mastercard's protocol, but before doing so we include some extra details common to both.

IAD: This hex-string follows the defined format set out in the EMV standard, but the details are proprietary; see *Visa Contactless Payment Specification* and *Visa Mobile Contactless Payment Specification* [10]. The IAD, in combination with the transaction data, is used by the bank/payment networks for anti-fraud checks. We discovered details of the IAD via investigation of cards and our disclosure processes with Visa, Apple and Mastercard, we give details in Section IV-C.

Cardholder Verification: EMV transactions with NFC cards remain fully contactless as long as certain spending limits are not reached: a limit per transaction (e.g., £45 in the UK) and/or cumulative daily limits (e.g., €150 in the EU); if either is reached, then normally the transaction would require proof of "user presence", i.e., a *Cardholder Verification (CV)* mechanism is enforced.

The Cardholder Verification Method (CVM) *list* informs the terminal of a set of rules for performing *Cardholder Verification* supported by the card, as well as the conditions in which these rules apply. For plastic cards, this is generally done by requesting that the card's Personal Identification Number (PIN) is input into the terminal (this PIN is sent encrypted to the bank for verification). On mobile devices, the CV can be done on the device, called Consumer Device Cardholder Verification Method (CDCVM), i.e., the user's fingerprint is scanned by the mobile app. Importantly, this allows the reader to accept contactless payments above the normal contactless limit. We note that the AIP indicates if CDCVM may be possible, not that it has been used.

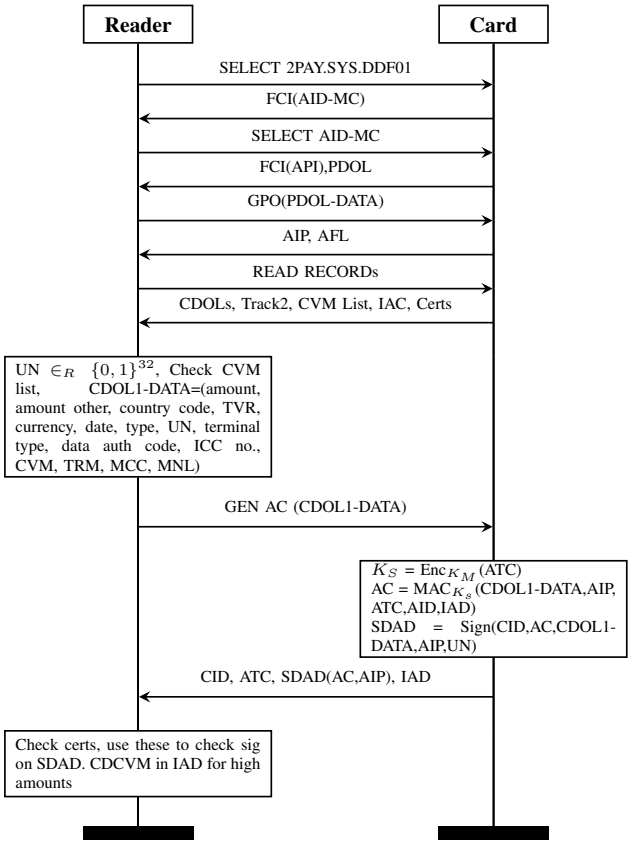

Fig. 1. MasterCard's PayPass from EMV standard & observed traces

"Tap-and-PIN": The way of requesting the PIN can differ from country to country (i.e., residing country of the terminal, issuing country of the card and combinations thereof). For instance, in the UK and Singapore, when using a UK-issued card, an "over-the-limit" transaction asks for the card to be inserted into the terminal and for the PIN to be used. Yet, in Spain, France, Switzerland, Norway, and others, the card does not need to be inserted in the terminal, but the user is asked to type in the PIN (or confirm via a button). We refer to the latter type of PIN-request as Tap & PIN mode. Investigating Tap & PIN cards from Romania and non-Tap & PIN cards from the UK, we found that non-Tap & PIN cards would stop the transaction for any amount over the limit, whereas Tap & PIN cards would continue, requesting the reader perform CV.

To the best of our knowledge, we are the first to point out that there are two types of EMV cards, although some past

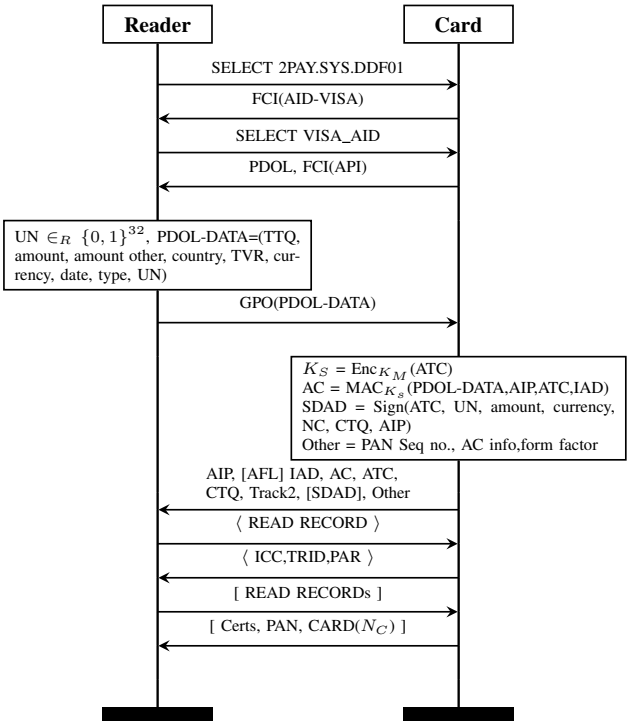

Fig. 2. Visa's PayWave protocol. Brackets indicate optional messages.

authors have presented attacks that only work against Tap & PIN cards [2], [7] and others have presented attacks that only work against non-Tap & PIN cards [11].

2) Visa's Protocol: The version of Visa's protocol, as per the standard and validated by our traces, is shown in Fig 2. Unlike Mastercard, the list of data needed for the transaction (e.g. amount, UN, etc.) is returned in answer to the second SELECT message. The function of the GEN AC and the GPO messages in the Mastercard protocol is merged into the GPO message in Visa's protocol. Checks on the SDAD and AC remain the same. While the MCC is not sent to the card, it is sent securely from the reader to the bank and payment-networks (i.e., Visa), for anti-fraud checks and fees [12].

The type of Cardholder Verification supported or performed is signalled through a number of different EMV data elements throughout the transaction, in particular a mobile Visa transaction uses the Terminal Transaction Qualifiers (TTQ), Card Transaction Qualifiers (CTQ) and AIP.

The *Terminal Transaction Qualifiers (TTQ)* inform the card of the online and CVM options that the terminal supports; of relevance are the bits flagging support of "EMV Mode" (byte 1 bit 6), "offline data authentication for online transactions" (byte 1 bit 1) and "CVM required" (byte 2 bit 7). Offline data authentication for online transactions is a feature used in special-purpose readers, such as transit system entry gates [13], where EMV readers may have intermittent connectivity and online processing of a transaction cannot always take place. In such cases, offline verification is performed and the payment is processed once the terminal is back online.

The *Card Transaction Qualifiers (CTQ)* are a set of options which determine what type of CV can/should be performed at the point of sale. The allowed options are decided by the bank issuing the card and are programmed at the time of issuance. The "CDCVM performed" bit (byte 2 bit 5) is of interest – it tells the terminal that on-device CV has been performed.

Unlike for Mastercard, we saw both online and offline mode in the Visa traces. If the "offline data authentication for online transactions" flag was set by the reader then the card would report extra records in the AFL field and would send the SDAD; the fields in square brackets in Fig 2. For Visa, the mobile devices we tested used tokenization [14] to obscure the account details (done by the read record in angle brackets in Fig 2), whereas plastic cards do not do this.

B. Over the Limit Attacks Against Tap & PIN cards

Two attacks have demonstrated how user authentication can be bypassed for high-value transactions with Tap & PIN cards. Galloway and Yunusov [7] show that, for high-value Visa transactions, a MitM attacker clearing the TTQ bit, which requests user authentication, leads to a high-value transaction being accepted without a request for the PIN. This shows that the TTQ used by the card is not being authenticated by the reader or EMV back-end.

Basin *et. al.* [2] present an attack against contactless Visa plastic cards, in which a MitM attacker flips CTQ bits, making the terminal believe that CDCVM was performed on the device when in fact it wasn't. This too leads to a high-value Visa transaction being accepted without the reader asking for a PIN. The authors of [2] state that their attack is "possible because no cryptographic protection of the CTQ is offered". While the lack of CTQ authentication is true, this is not the root cause. We show in sub-section IV-C that the IAD generated by the plastic card in their attack would have a Visa "plastic-IAD" format [10] which, if checked by the payment network, would reveal that the device is not capable of CDCVM authentication, and so the transaction should be rejected. Therefore, the attack of Basin *et. al.* [2] is due to missing checks at the EMV back-end rather than a flaw in Visa's protocol.

We observed some discrepancies regarding how the TTQ-AC relationship is presented in the above mentioned attacks. Galloway [7] claims that for Visa cards, the AC does *not* contain the TTQ, based on information from the *Visa Contactless Payment Specification* [15], which is a proprietary specification. Basin *et. al.* [2] claim that the TTQ *is* in the AC, based on EMV Book 2 [16], and explain that this is why their formal model does not identify the Galloway attack. EMV Book 2 states that it is *recommended* that the whole Processing Options Data Object List (PDOL) be included in the AC (which would include the TTQ in this case), but the *minimum set of data elements* they specify *does not* include the TTQ. Therefore, whether the TTQ is included in the AC or not (and if the attack of Galloway [7] is due to missing back-end checks or a flaw in the protocol) is unclear.

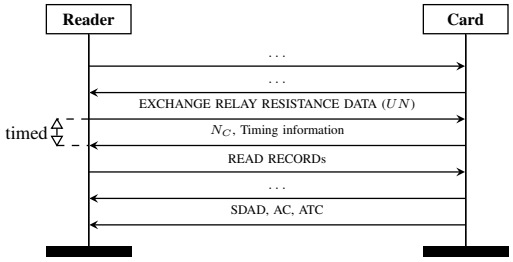

Fig. 3. Mastercard's PayPass-RRP Protocol – PayPass with $ERRD$

C. Visa Relay Protection Protocol

Visa proposes a relay-counteraction measure [3], [17], which we call the *VISA-L1* protocol. This protocol is based on two ideas. First, it requires the card to use a random 4-byte Unique Identifier (UID) in each run of the protocol (random UIDs are common in RFID devices). This means that the UID now functions as a nonce, and is referred to by Visa as the `L1SessionParameter`. This is sent by a card to the reader as part of the Level 1 anti-collision process (see Appendix A). The `L1SessionParameter` is then tied into the Level 3 of Visa's PayWave protocol. While Visa's patent and current documents do not specify how the Level 1-to-Level 3 binding must be done, we understand from conversations with Visa that the "EMVCo NextGen" specification will specify that the `L1SessionParameter` be added to the SDAD, alongside the normal Level 3 EMV data that the SDAD contains. If the UID received at Level 1 and Level 3 do not match then the transaction is rejected as a possible relay.

Visa's Relay-Security: In the specification documents [3], [17], the security argument relies on the difficulty of setting the UID to a particular value, especially with off-the-shelf devices such as mobile phones. Thus, a proxy card used in relaying to impersonate the real card would fail to produce the right messages to the legitimate reader.

Through our conversations with Visa Research, there is another implicit security argument, common in the RFID field: relaying at Level 1 is harder than at Level 3, because ISO 14443-4 framing is more restrictive than at the EMV level. So, relay protections may be more effective at Level 1 than Level 3. However, we note that if the attacker can set the UID of the proxy to equal the UID of the card, then Visa's defences will no longer work , because there is no dynamic/fresh reply by the reader based on said UID and there is no distance bounding used in the VISA-L1 protocol.

D. Mastercard Relay Protection Protocol

Mastercard's PayPass-RRP (shown in Fig. 3 and described in EMV Book-C2 [1], Sections 3.10, 5.3 and 6.6) is a direct extension of the PayPass protocol, in which a timed nonce-exchange at Level 3 is used in order to detect relay attacks. PayPass cards indicate they support the protocol with a `AIP` of 1981; a PayPass-RRP reader then sends an Exchange Relay Resistance Data ($ERRD$) command that contains the "Terminal Relay Resistance Entropy". This is the same reader-generated UN nonce sent in PayPass inside the GEN AC. The $ERRD$ response contains (1) the nonce returned by the card and denoted in Fig. 3 as N_C, (2) three timing estimates from the card, denoted in Fig. 3 as *timing info*, i.e., the minimum and maximum expected time for the card to process the $ERRD$ command and an estimate of the round trip time (RTT). All these values are signed in the Signed Static Application Data (SSAD), which the reader should check.

If the message RTT is smaller than the maximum listed in the timing data, then the $ERRD$ phase finishes and the protocol continues. If the RTT is larger than the maximum time three times in a row, then the reader stops the transaction as a suspected relay attack. If a terminal has done a PayPass-RRP check and it passed, then the `TVR` should be set to 0000000002. We will use "RRP" for the whole payment protocol by Mastercard (PayPass-RRP).

III. Threat Model

Our threat model is that of an active Man-in-the-Middle (MitM) adversary, who can also relay. The attacker operates in an environment where: (1) the banks/issuers/payment networks are honest; (2) the EMV terminals are honest; (3) cards can be compromised, except for the card that the attacker is trying to relay in the current attack (i.e., we do not consider attacks such as distance fraud or terrorist fraud).

Formal-Verification Adversary: Our attacker is modelled as a Dolev-Yao attacker [18] allowing for corrupt cards and an unbounded number of sessions. For proximity-checking, we follow the state of the art formalism of Mauw *et. al.* [5], where distance and timing are safely abstracted into event-ordering on traces, and we are only interested in MitM-security (i.e., not distance fraud or terrorist fraud).

Practical Adversary: Our practical attackers use, for relaying or other MitM manipulations, Commercial Off The Shelf (COTS) equipment, i.e., commercial, relatively non-expensive, easy-to-use hardware or software such as mobile phones (rooted or not). More specifically, our practical attacks do not rely on extensively modifying firmware on hardware or building new hardware (for relaying or other MitM attacks), and our practical adversaries stop at application-level development/manipulation on COTS devices. This is the same type of attacker that Mastercard and Visa aim to stop with their proposals. No current proposal for relay protection for contactless EMV aims to stop specialist, expert-built relay/MitM equipment (e.g., [19]).

There are fast and effective, purposely built hardware-based relays in other domains such as remote car-unlocking [19], as well as solutions [20] designed specifically for the physical-layer (e.g., bespoke modulation schemes) to combat such efficient, hardware-based attacks. Our threat model does not include hardware-based EMV relays that operate at Level-1 (or the physical-layer) and such relays might be able to compromise our proposed solution. In fact, in Section VI, we give certain timing measurements for an implementation of our `L1RP` protocol, mention the bottlenecks observed, and leave as an open question the possibility of successful hardware-based relays against current contactless EMV technologies.

IV. Mobile-Payments via Transport Mode

This section presents our results from experimenting with Apple Pay Express Transit (known as Express Travel in Europe) and Samsung Pay "Transport card". We refer to these two systems as "Transport mode". The transport mode on these phones is a convenience feature, which allows a user to pay on certain transport networks *without* prior authentication to the device (fingerprint, face ID or passcode), by simply tapping the phone on the EMV reader of the transport network.

Apple Pay's transport mode is available in London (TfL), New York City, Portland, Chicago, Los Angeles, Washington, Beijing, Shanghai, Hong Kong and Japan [21]. Samsung Pay's transport mode is only advertised to work in London (TfL) [22]. Google Pay allows, by design, for certain small-value transactions without user authentication[2] and does not have a dedicated transport mode.

A. Setup and Data Collection

Hardware: As target devices we had two Apple products (an iPhone 7 and an iPhone 12, running iOS 14.4.2 and two Samsung products (a Samsung Galaxy S9 Edge and S10 running Android 11. We tested our findings on a number of commercially available EMV terminals: iZettle v1, SumUp, and SquareUp readers.

Preparation Stage: We collected transport mode transaction traces between a locked phone and TfL ticket gate readers at Clapham South, Clapham Common, Balham and Epson London Underground stations. For this we used a Proxmark RDV4 fitted with the high frequency antenna, running standard firmware[3]. The Proxmark was chosen both due to its versatility and excellent ISO 14443-A support, as well as its small form factor, which meant we could carry out the sniffing experiments without drawing too much attention. The Proxmark was connected via USB to a Linux laptop in order to issue the sniffing commands.

Analysing the sniffed traces, we noticed that an extra, static 15-byte message is sent before the ISO 14443-3 Wake-UP command, Type A (WUPA) [23], which we refer to as the *Magic Bytes*.

B. Visa Apple Pay Express Transit Replay & Relay Attack

We found that by replaying the *Magic Bytes* to an iPhone, Apple Pay would unlock and allow us to start a transaction for both Mastercard and Visa, even if no authentication (PIN, FaceID or TouchID) has taken place. However neither would initially let us then complete an EMV transaction.

By examining the Visa traces we found that in order to continue with the transaction in transit mode, we have to make sure the TTQ, contained in the GPO, has the *Offline Data Authentication (ODA) for Online Authorizations supported* bit set (byte 1 bit 1) as well as the *EMV mode supported* bit set (byte 1 bit 6). The former flag indicates to the device that the reader may be offline, but that the device should use online mode anyway.

[2]https://support.google.com/pay/answer/7644132

[3]https://github.com/RfidResearchGroup/proxmark3

By setting these flags in a relay we found we could relay transactions between a locked iPhone and any of our shop readers, therefore, allowing the attacker to wirelessly pick pocket money from the iPhone's owner. The attack may also be used to extract money from a stolen, locked iPhone. We note that it would be possible for Visa to try to authenticate the TTQ and so limit the attack to just shop readers that have the Online/Offline mode flag set, however, as shown by [7], Visa do not authenticate this field.

Attack Details: This vulnerability has a Severity Score of 7.1 and a High Severity rating (based on the Common Vulnerability Scoring System v3.1 [24], [25]).

For carrying out the attack, we used a Proxmark RDV4, with small modifications to the device firmware (details to follow) which acted as a reader emulator, the EMV readers mentioned above and an NFC enabled Android device which acted as a card emulator (Nokia 6 TA-1033, running Android 9 Pie, in our case). A device to run the relay server was also needed; for our experiments we used the same laptop as above.

To carry out the attack a Proxmark acts as a terminal emulator, and it is connected to a laptop via USB. The Nokia phone acts as a card emulator, using the `CardEmulator` app from [26], and communicates with the laptop via some network (cellular or WiFi). We modified the Proxmark firmware such that, when emulating an ISO 14443 type A reader, it would first send the *Magic Bytes*, wait $8ms$ for the iPhone to process the message, then send the WUPA command. The rest of the ISO 14443 protocol is run according to the standard, after which the EMV protocol can begin. This step only involves the iPhone and the Proxmark, and no messages are relayed in this part of the attack.

The relay server runs a Python script which handles the communication with the Proxmark, via the `pm3` client, and the communication with the `CardEmulator` via a socket. It relays the messages between the Proxmark and the `CardEmulator`, and can modify them in-flight, as needed.

Once the iPhone is in the ACTIVE state (as described in ISO 14443-3), we can start the interaction between the card emulator and reader. They will complete the standard ISO 14443 activation protocol.

The EMV protocol then starts, and the messages are relayed between the EMV reader and the iPhone. A diagram of the relay can be seen in Fig. 4. At the point in which the reader provides the transaction information, as part of the GPO message, in order to continue with the transaction, we *set* the *Offline Data Authentication (ODA) for Online Authorizations supported* bit (byte 1 bit 1), as well as the *EMV mode supported* bit (byte 1 bit 6) in the TTQ – we denote the modified message as GPO' in our diagram. The EMV protocol continues as normal. We would like to highlight that as part of the answer to the last READ RECORD command the phone sends, among other details, the Card Authentication Related Data (CARD), which contains the card nonce (N_c) and a *copy* of the CTQ. Once the last reply is received by the reader, the transaction is successfully completed - the iPhone shows a tick and displays 'Done' in the user interface (UI), and issues its

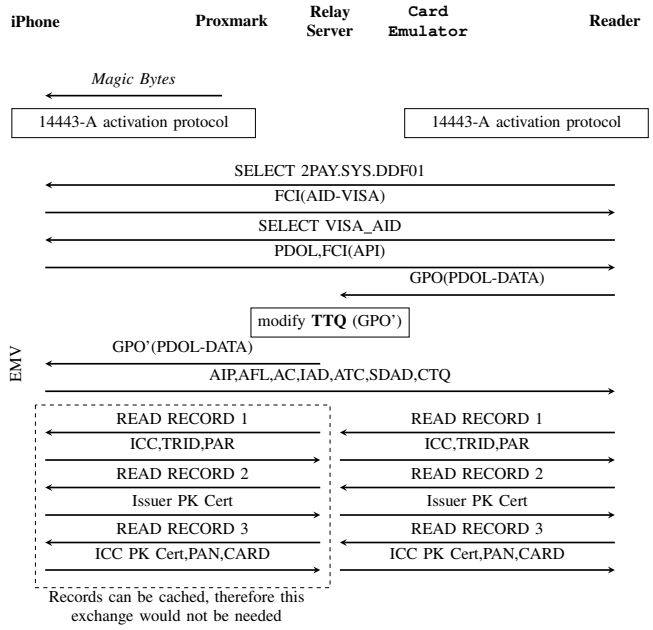

Fig. 4. Communications diagram for the Apple Pay Express Transit attack

trademark sound, and the reader shows the payment has been received.

If the attacker can interact or eavesdrop the iPhone before the attack then the READ RECORD commands can also be cached, in which case the Proxmark no longer needs to relay the requests for these to the iPhone. The replies can be cached on the relay server or, ideally, on the device acting as the card emulator. If the replies are cached the UI does not show the transaction as completed; instead, it displays 'Try Again' and a red exclamation mark. There is also no audio cue. However, the reader shows the payment has been successful, and the transaction can be confirmed by unlocking the iPhone and checking the Apple Pay transaction history.

The reader is running in online mode and is not expecting the SDAD from the iPhone, which is only sent because we flipped a bit in the TTQ, i.e., the iPhone is running in DDA mode and the reader is running in EMV mode. We could remove the SDAD from the DDA message to make it look like a EMV message, however we found that, if sent, the reader will just ignore the SDAD and does not check it.

Relay Success Rate: The full relay has a success rate of approx 40% (8 out of 20 trials). This success rate is achieved by a slight modification in the attack flow, whereby when the first READ RECORD command is issued, we continue the phone-side only messages exchange with the two subsequent read commands. We then relay them to the reader, as it requests them (as opposed to forwarding each READ RECORD command when received from the reader). We point out that the success rate is not due to any Apple protection mechanism, but due to the timing of relaying the messages, and for the cached READ RECORD variant, the success rate is 100%.

Over the CVM Limit Payments Escalation: The above described attack works for transactions *under* the contactless payment limit. We were able to escalate this attack to work for any amount. We could bypass the contactless limit by *setting* the CDCVM performed bit in the CTQ (bit 8 in byte 2), *as well as* setting the same bit in the CTQ copy of the CARD field. We successfully tested this with amounts up to £1000. The code for the proof of concept is open sourced and available in [9] along with a video of the attack being performed.

Disclaimer: For ethical reasons, we used our own personal cards while carrying out this research and we do not publish the value of the *Magic Bytes*.

Mastercard: We have also investigated whether this attack works when using a Mastercard with Apple Pay Express Transit. In order to complete a transport transaction, we found that the Merchant Category Code (MCC), which the EMV reader sends, needs to map to transportation services [27]; we confirmed it works with MCC `4111` (local commuter transport) or `4131` (bus lines). However, the MCC is authenticated in the Mastercard protocol (it is included in the SDAD), and therefore this modification is detected and the transaction rejected. This check on the MCC is the reason why we cannot relay Mastercard transactions from a locked iPhone to a shop reader, although relays to other transit readers are possible.

C. Investigation of Mode-Change in Samsung Pay

We investigated Samsung Pay Transport Card, the equivalent of Apple Pay Express Transit, on a Samsung Galaxy S9 Edge and Galaxy S10, running Android 11, with all updates applied. We found that the transaction logic of the Samsung phone differs from the iPhone; when it comes to choosing whether a transaction should be executed via its transport mode or through the normal mode. The Samsung phone did not require the presence of the *Magic Bytes* to respond to EMV commands. Samsung's solution relies on one of the suggested mitigations we had for Apple: small/zero value payments. If the transaction value is 0, the phone considers this to be a transaction with the public transport infrastructure and executes it through the transport card (no authentication required). A larger amount is considered to be a normal transaction, and requires authentication. If the fingerprint (or Samsung Pay PIN) is not provided beforehand, the phone replies with a message to signal that the conditions of use have not been satisfied and stops the transaction.

We could activate the Samsung Transport card by replaying a TfL trace to it. However, when trying to relay a non-zero value transaction between a real EMV reader and the phone, the transaction was always cancelled by the phone, and a message that authentication to the device is required was displayed. Changing the amount of the transaction while relaying (in the same way we changed the TTQ when relaying the iPhone) resulted in the transaction reaching the end of the protocol on the phone, but being declined by the EMV reader.

As it is clear that the transaction is cancelled by the phone if any non-zero value is present in the *authorised amount* field of the GPO message (which is used for purchases), we also experimented whether the same occurs for the *other amount* field, which is used for cashback. Here we found that we *could*

complete a transaction with a locked Samsung phone (and therefore had an AC generated) with some value in the *other amount* field, as long as the *authorised amount* value was zero.

Plans have been announced to allow cashback without a purchase on contactless cards[4] transactions from locked Samsung phones. Unfortunately, we were unable to acquire readers that support the cashback functionality, in order to test this. We hope to be able to try this attack in future.

D. Investigating the IAD

To find out how CDCVM was authenticated we carried out a byte-by-byte comparison of mobile device transactions with and without CDCVM (transport mode). Looking at Mastercard we found that the only differences were in the UN, AC, SDAD, and the IAD. The AIP was the same whether CDCVM was used or not, always indicating that the device supports CDCVM, i.e., the AIP does not indicate that device has performed user authentication. We would expect the UN, AC and SDAD to be different for each run because they are designed to be fresh values, therefore this suggests that the IAD indicates if CDCVM was successfully performed on the device.

We tried a MitM attack that inserted the IAD from a user authenticated transaction into a transaction with no user authentication and the transaction was rejected, indicating that the IAD is authenticated. Mastercard later confirmed to us that the IAD is included in the AC and this is checked.

For Visa plastic cards, we found that there are multiple IAD formats used. We saw in our transactions IADs with "Format 0/1/3", for example: 065C0A03A00000. The general description of an IAD is defined in EMV Book 3 [28]. Bytes 5-7 of the IAD are the Card Verification Results (CVR), which indicate whether a second GEN AC was issued, and the type of AC returned, among other things.

On mobile devices we saw 32 byte IADs starting:

Apple (CDCVM) : 1F4A5732A0000000001003...
Apple (transport) : 1F4A5760A0000000001003...
Google Pay (CDCVM) : 1F4346512000000000000000...
Samsung (transport) : 1F4363002000000000000000...
Samsung (CDCVM) : 1F4346422000000000000000...

While the format of these longer IADs is not public, we see that the first byte of the CVR, highlighted in red, always reflects whether or not CDCVM was used, therefore indicating to the EMV back-end if CDCVM was used on the device. During the disclosure process Visa confirmed this, and that these bytes were in the AC, but that Visa does not check them.

In summary, Mastercard use the IAD to authenticate the use of CDCVM. From the IAD, Visa could determine if the transaction was with a device capable of performing CDCVM, and if CDCVM was used, and these could be authenticated against the value in the AC, however this check is not currently performed. If this check was performed then the Visa over the limit attack against the iPhone would not be possible.

[4]https://www.moneysavingexpert.com/news/2020/10/shoppers-to-be-able-to-get-cashback-without-buying-anything-unde

E. Responsible Disclosure

The details of this vulnerability have been discussed with Apple and Visa. In a meeting with the Apple Pay security team, they stated that they believed that this was a serious security issue, but they believe Visa would be responsible for allowing this IAD-unchecked transaction to go through. Explicitly, they stated that Visa should identify iPhone transactions, check the CDCVM status in the IAD, and that iPhone transactions without CDCVM should only be allowed when the MCC code indicated transit payment.

We suggested restricting transactions to low or zero value, when Apple Pay is unlocked with the *Magic Bytes*. Zero value is in line with the operations/transactions needed by TfL, which calculates the correct amount to charge travellers at the end of the day based on all their trips, but Apple stated that they support transit systems that require non-zero payments, so they do not want to put any bound on the payment value in transport mode. Apple did not pay a bug bounty, even though they advertise $100,000 for bypassing a lock screen, and our attack bypasses the Apple Pay lock screen.

We have also discussed this attack with Visa, who pointed out that this attack only affected Apple Pay, and suggested Apple were best placed to fix the issues. Visa also stated that back-end anti-fraud checks weregenerally applied, when needed. So, if this attack was to raise fraud-alerts, they claim, it would be eventually stopped. That said, we performed our attack multiple times, on large values, from the same card, and we were never blocked and flagged for fraud. Until either Apple or Visa implement a fix, we recommend that iPhone owners disable transit mode for Visa cards.

F. Comparison with Existing Attacks Over the Limit Attacks

The attack presented by Galloway [7] is used to perform over the CVM limit transactions without Cardholder Verification on plastic cards and involves *clearing* the CVM bit in the TTQ – i.e. the reader will believe that no CDCVM is required. For over the limit transactions on Apple Pay Express Transit, only clearing the CVM bit in the TTQ (without the CTQ flips) results in the transaction *failing on the reader side*, i.e., this attack does not work against Apple Pay.

The attack presented by Basin *et. al.* [2] is a different approach to achieving over the CVM limit transactions without requiring Cardholder Verification, on plastic cards. It involves *clearing* the online PIN verification required bit and *setting* the CDCVM performed bit in the CTQ – i.e. tricking the EMV reader into believing the card has performed CDCVM. Our Apple Pay Express Transit CVM limit escalation also targets the CTQ. However, the online PIN verification bit is not set in our case. We target the CDCVM bit in CTQ in *two* places: it first appears in the response to the GPO message (under tag `9F6C`), *and* in a record template, as part of the CARD (tag `9F69`). The CDCVM bit needs to be *set* in both of these EMV tags, otherwise the transaction is not successful.

As a side-note, we confirm the Galloway attack *still works against Google Pay* (Pixel 5), two years after being publicised [7]. We found that the Basin *et. al.* attack did

not work against Google Pay: the combination of over the limit transaction amount and set CVM bit in TTQ results in the phone terminating the transaction after it receives the GPO message, with the code `6986` (Command not allowed), and therefore the CTQ is never sent. We conclude that Google detects and rejects any Visa transaction that requests authentication in the TTQ unless the phone has been unlocked. Unfortunately Apple Pay does not have this defence.

G. Formal Verification

To analyse these protocols we use the Tamarin prover [18]. This is a verification tool that supports symbolic/Dolev-Yao analysis [29], [30]. Tamarin models are transition systems over a multi-sorted term algebra, operating on the semantics of multiset rewriting logic [31]. Security properties can expressed as lemmas about the labels on the rewrite rules. Tamarin can then automatically either prove that a security proprieties holds or provide a counter example, i.e., find an attack.

1) Verifying Visa in Apple & Samsung Pay: We use formal verification to show that our attack is exhibited on the EMV specification of Visa used inside an Apple Pay app but not inside the Samsung Pay app. We show, formally, that the countermeasures proposed to Apple and Visa stop the attack. Finally, we show that our attack cannot be completely counteracted by any/all of the countermeasures, i.e., that it is still possible to relay to terminals that share the same MCC.

We endeavoured to create this model to account for a modular treatment of the countermeasures we discussed, e.g., Apple and Samsung Pay differing only in the answers to GPO commands based on value inside this message, etc. The Tamarin file can be found in Mobile_Visa.spthy in [9].

Tamarin Model for Mobile Visa: We used as starting point the Tamarin models for contactless Visa "plastic cards" by Basin *et. al.* [2]. Unlike Basin *et. al.*, we have one single model which contains: (a) both EMV transaction-authorisation modes ("DDA" – with SDAD, used in mobile transport mode, and "EMV" – no SDAD, used in non-transport mobile); (b) transaction-values above and below the limit ("high" vs "low"). Most of our Tamarin rules for card, terminal and bank are similar to the one in Basin *et. al.*, but other extensions/-modifications are necessary as we explain in the following.

Terminals. A "`Create_Terminal`" rule generically creates our terminals as follows. A terminal can be a "transport" vs "non-transport" terminal, and if it is the former it sends the "magic bytes" we observed TfL to send. Terminals can send "zero", "low" and "high" values. Terminals have an MCC value, which in the model is as broad as "transport"/"non-transport" MCCs. To encode the Apple Pay business model, our protocol rules indeed allow transport terminals to send any value (see `Terminal_Sends_GPO_AnyValue_AnyMode_Visa` rule). In this rule, we impose however a realistic restriction that non-transport terminals cannot accept zero values (i.e., see `_restrict(NonTransportNonZero(mode,value))`.

Cards/Mobile Apps. We create rules for cards/apps as generically and modularly as possible, with the same transition-rule applying to both behaviours wherever possible. Before the app actually responds to the GPO, we have a `ComputeCVR` rule firing that implements the mobile-app logic of judging if CDCVM is needed based on the "magic bytes" being received or not (i.e., perceived app operation mode) and value (high/low) sent in the GPO command.

We have two GPO-responding rules separated by the `Handset('samsung')` vs `Handset('apple')` and otherwise the facts produced by this `ComputeCVR` rule; these two GPO-responding rules are the only way we differentiate between Samsung and Apple Pay w.r.t. EMV-card behaviour; i.e., in the case of Samsung, we impose a restriction `_restrict(ZeroOnly_in_nonAuthen_Transport (CDCVM_noCDCVM, perceivedAppMode, value))` to allow only zero payments when in the `ComputeCVR` rule it judged it is in transport mode.

Unlike the Basin *et. al.* model, in the GPO-responding rule, the cards create a more detailed IAD. Concretely, e.g., in the Basin *et. al.* model, the IAD for Visa EMV mode was `IAD = <'IAD', CID>`; ours is `IAD=<'IADmobile', CID, CVR, format>`, and this additionally denotes that the CVR (denoting if CDCVM was performed is included in the IAD), and "format" specified if the card believes it is operated in transport or non-transport mode. This is entirely in line with Visa IAD formats for mobiles. Also, in our model, we add that the MCC of the terminal is finally sent to the issuing bank along with the whole transaction on a secure channel.

The Bank. We implement three transaction-processing rules, accounting for the different possible behaviours: (case1) as in the models by Basin *et. al.*, in this case the bank does not check the CVR and format values inside the IAD, (case2) the bank does check the CVR and format values inside the IAD, but does not cross-check these against the MCC; (case3) the bank checks all of the IAD, MCC and transaction data.

Verification. In the model, we add numerous sanity-check lemmas to show that all and only faithful behaviours w.r.t. to Apple/Samsung, transport/non-transport, and values are present. We then prove/disprove the following lemmas: respectively meaning:

1) the Apple-Pay attack is found:
   a) via falsifying the "all-traces" payment-security lemma1.a, for "case1" of the bank-checks above, where the bank does not check the IAD;
   b) via proving an "exist-trace" lemma1.b
2) Samsung Pay does not suffer from the mode-abusing payment attack (via proving an "all-traces" payment-security lemma2, quantifying over traces of `Handset('samsung')`)
3) either of our two countermeasures stops the Apple attack: i.e., the bank checking the CVR (lemma3), and the bank checking the MCC against the IAD-format (lemma4).
4) it is still possible to relay a transaction from a transport terminal to another transport terminal, if they share the same MCC (lemma5)

Note. Checking the format of the IAD (i.e., case2

above) is enough to stop the "CTQ-change", "Tap & PIN" attack in [2]. This is shown indirectly by our lemma3 above. We also show this for the original model Visa_EMV_High.spthy from [2]. I.e., the authentication for the bank (i.e., `lemma auth_to_bank_minimal`) holds if the IAD format is checked. In the GPO-processing rule, the TTQ info received by the card now dictates the IAD format, and in the rules relating to the bank processing the CVM (for the case of online/not-online PIN) we added IAD format checking (see lines 425, 749 of the amended file Visa_EMV_High_BasinEtAl.spthy in [9]).

2) Verifying Mastercard in Apple & Samsung Pay: We use Tamarin to verify that no similar attack is possible against express transit mode for Mastercard on Apple Pay. The Tamarin file can be found in Mobile_Mastercard.spthy in [9].

We detail slightly less in this case, building on the previous section. Like for Visa, we base our model of Mastercard on the work of Basin *et. al.* [2], to this model we add a) the more detailed IAD that encodes if the device used the user authentication (CDCVM) or not, b) that devices may indicate in the AIP that the device supports CDCVM but that a device might not use it, c) we add the Merchant Category Code (MCC). Our experiments and conversations with Mastercard's security team indicate that (unlike Visa) all of these values are actually checked. We also make some simplifications to the model in [2]: i.e., we only model Mastercard with a SDAD, as we have not seen any support for transactions without this across any of our devices or readers. Unlike for the Visa model, we only consider two payment amounts, a low and a high value. For Apple Pay Express Transit, we add that devices may function without the device authentication if the "magic-bytes" action is present in the trace and if the MCC code indicates that the terminal is a transport reader.

Using Tamarin we are able to prove that: for uncompromised Mastercard Apple Pay devices, if the bank accepts a high transaction amount then the device must have used CDCVM user authentication and we can further show that this is based on Mastercard checking the IAD, indicating that this is an important part of Mastercard's protocol.

To show that the Visa Apple Pay attack is not possible against Mastercard, we show that the bank will only accept a non-CDCVM transaction from a terminal with a transport MCC code. This means that relay attacks using transit express mode are limited to only relaying to other transit terminals.

V. Visa's Level 1-Relay Protection

As stated in their attacker model, Visa's solution relies on the inability of the attacker to change the UID of a card or mobile phone, which they refer to as `L1SessionParameter`, and the difficulty of relaying the Level 1 messages due to their timing constraints. However, setting a desired UID on some mobile devices is possible [32], if the device is rooted. This has been introduced in Android 4.4, as *host-based card emulation*, and allows an app to emulate a card (or NFC tag) and talk directly to a NFC reader [33]. While this is a departure from Visa's attacker model, rooting a phone is not a complicated task in 2021, with plenty of resources available and tools which take care of the more "technical" steps of the process. We have tested a rooted Nexus 5 phone, which has the Broadcom BCM20793M NFC controller, running its stock firmware (Android 4.4) and running CyanogenMod 14.1 (Android 7.1.1). On both versions we were able to successfully set any UID we wanted by editing the NFC configuration file.

By building on the work of [26], we modified their Android relay apps to add an extra step before any EMV messages were exchanged. We ran the `CardEmulator` app on the Nexus 5 phone and used a Nokia 6 phone to run the `TerminalEmulator` app. A server forwards data between the apps (no change to the original from [26]).

On the `TerminalEmulator` app we added extra functionality which allows us to retrieve the UID of the card the phone is in contact with. This is done through retrieving the *ID* of the `Tag` object that was discovered by the `ACTION_TECH_DISCOVERED` intent, and sending it to the server, which will then forward it to the `CardEmulator` app.

In the `CardEmulator` app we receive the UID and set it as the phone's UID by:

- Remounting the `system` partition with read and write permissions;
- Replacing the `NFA_DM_START_UP_CFG` parameter in the `/etc/libnfc-brcm-20791b05.conf` configuration file such that it includes the UID:
  - we add 6 to the first byte of the config parameter, which holds the length of `NFA_DM_START_UP_CFG`, as we will be adding 6 more bytes;
  - at the end of `NFA_DM_START_UP_CFG`, we add the NCI parameter `LA_NFCID1` (`0x33`), which means the UID is declared in this configuration parameter, the length of the UID (`0x04`), and the UID itself;
- Restarting the NFC service.

The complete set of steps for setting the UID on the phone takes approximately 181ms. After this, the EMV level relay can proceed as normal. We confirmed with a Proxmark that the UID of the Nexus phone was equal to the UID of the relayed card. Fig. 12 in Appendix B presents the overview of the communication between all the devices involved. Our modified apps are available in [9].

With this attack we can break Visa's relay protection protocol. When the `L1SessionParameter` is sent signed as part of the EMV protocol, the EMV reader will decrypt it, and it will match the UID of the phone. We can achieve this both because we can set a phone's UID as desired and because, although there is an overall timeout for the transaction (the EMV specification says 500 ms [34]) there is no round-trip timing measurement within Visa's protocol, which means the card may wait in the PROTOCOL state (before the EMV protocol starts), for the SELECT 2PAY.SYS.DDF01 message from the EMV reader, as the transaction timing is enforced by the EMV reader with which we start interacting *after* the UID change has happened.

VI. Level 1 and Level 3 Timing in EMV

We experimented with the reliability of timing at Level 1 and Level 3, and therefore the feasibility of relaying at these two levels. We experimented with several commercial EMV cards (Visa and Mastercard), as well as aprototype, test prototype PayPass-RRP card, bought from a vendor called ICC Solutions. We found no commercial/bank-issued PayPass-RRP cards.

We show that timing exchanges at Level 1 are much faster than at Level 3, and show much less variation, and the Level 3 variation in timing is considerable for all EMV cards that we tested, and more so, for a *proprietary card that implements a test version of PayPass-RRP, we can relay at Level 3 with a standard relay program*. Overall, the variation at Level 3 across all cards, has led us to propose a new Level 1 protocol for relay protection in EMV, in Section VII, which improves on both Visa's and Mastercard's current solutions for relay counteraction.

A. Level 1 and Level 3 Timings for EMV Cards

We measured the Level 1 and Level 3 message round trip times (RTTs) for: (1) a Mastercard-RRP test card; (2) a "normal" commercial Mastercard; (3) a commercial Visa card. All the raw data, processing scripts and other programs we used are available in [9].

Hardware/Software Setup: We used an Advanced Card Systems ACR122u reader. A program acts as an EMV reader, prepares the correct sequence of challenges (ISO 14443 messages, or Level 2 Command Application Protocol Data Units (C-APDUs)) for a transaction with the card and, via the ACR122u reader, sends this to the card; the sequences were different for each card (PayPass-RRP, Mastercard, Visa), as each executes a different protocol. To obtain RTTs and other timings related to these exchanges, we placed a Proxmark in-between the reader and card, to sniff the transaction, and thus we obtained the full traces of the exchanges. This is discussed further in Appendix C.

Experimental Design: To capture the variations that occur when someone makes a contactless payment, we varied the yaw angle at which the card was placed in the field and the distance between the card and the reader. The distance was maintained by resting the card on a spacer. We tested at angles of 0°, 45°, 90°, 135° and 180°, and at distances of 5mm, 11.4mm, 21mm, 24mm, 27.4mm and 30.6mm. These are all realistic positions a card may be held in to make a payment. We took 20 measurements for each physical configuration, making a total of 600 tests. We note that, apart from the initial Level 1 messages, what we are actually measuring is the time from when the reader sends the C-APDU (at Level 2) until it receives the Response Application Protocol Data Unit (R-APDU) (also at Level 2) back from the card.

Generally, the reader sends a message/C-APDU and waits for a response/R-APDU from the card. We used the `hf 14a sniff` command of the Proxmark client, which sniffs the ISO 14443-A traffic and produces traces as per Fig. 5. Thus, we can calculate the reply time of a card

```
Start |   End | Src | Data (! denotes parity error)        | CRC | Annotation
------+-------+-----+--------------------------------------+-----+-----------
0     |  1056 | Rdr |26                                    |     | REQA
2260  |  4628 | Tag |04 00                                 |     |
13296 | 15760 | Rdr |93 20                                 |     | ANTICOLL
16964 | 22788 | Tag |3f af e3 54 27                        |     |
44144 | 54672 | Rdr |93 70 3f af e3 54 27 9a 70            | ok  | SELECT_UID
55860 | 59444 | Tag |20 fc 70                              |     |
67680 | 72384 | Rdr |e0 50 bc a5                           | ok  | RATS
77364 | 91252 | Tag |0a 78 80 70 02 20 63 cb b7 80 8b 30 | ok  |
```

Fig. 5. Excerpt from a Proxmark trace (timing in carrier periods $\approx 0.074\mu s$)

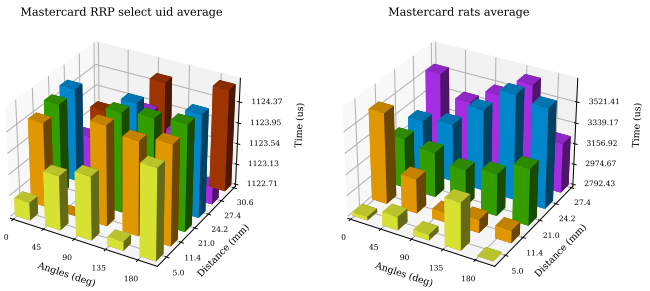

Fig. 6. Mean reply time for the 14443-A SELECT UID and RATS commands (note the different scales here)

to a reader command, both for Level 1 and Level 3 level messages. The reply time for one command is defined as $t = p\,(Rdr_{Start} - Tag_{End})$, where Rdr_{start} is the time when the reader starts transmitting its message and Tag_{End} is the time when the card stops sending its reply.

B. Timing All EMV Cards at Level 1

For Level 1 timings, we focused on the responses to the SELECT UID and Request for Answer To Select (RATS) commands that both occur after the anti-collision routine is complete. Note that the reply to the SELECT UID requires no computation from the NFC chip on the card and would be consistent with the behaviour of a nonce exchange if the card's nonce was prepared before the challenge was received.

Fig. 6 shows the average reply times for the Level 1 SELECT UID command and the RATS commands. The RATS command involves more processing and its times are longer, but still much shorter than those at Level 3. It can be seen that there is little change in the timings for the SELECT UID as we vary the angle or distance of the card. The standard deviation did not vary across angles or distances and was only 0.91µs and the time difference between the quickest reply and the longest is only 2.36µs. For the RATS, the "normal" Mastercard, had the longest average response time out of all the cards, of 3231µs, with a standard deviation of 286.50µs. So, for our experiments, timings at Level 1 appear relatively stable across all EMV cards.

C. Timing a RRP Test-Card at Level 3

For Level 3 timings, we focus on the nonce-exchange in the $ERRD$ command, although results were obtained for the other APDUs, so that they could be compared with the other cards that we tested (see Appendix C).

Our Mastercard-RRP test card implements up to three nonce-exchanges, as per the $ERRD$ command we described in Section II. In our tests, we sent all three nonce challenges,

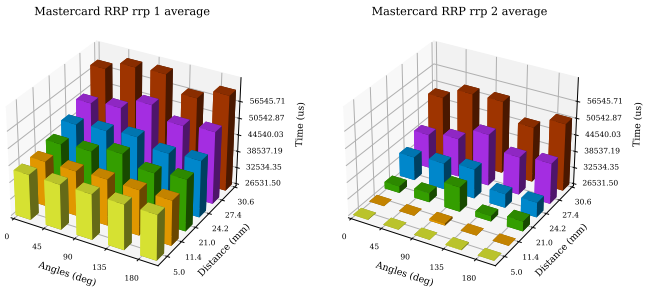

Fig. 7. Mean reply time for the nonce exchange commands implemented on our Mastercard-RRP Test Card

i.e., three C-APDUs, and measured the response times for each (labelled: rrp 1, rrp 2 and rrp 3). The average timings for the first and second nonce-exchange messages at the different heights and angles are shown in Fig. 7. The timings for the third nonce exchange are very similar to those for the second. The statistics for these three exchanges (mean/SD in µs) are: 53,000/13,170, 40,100/15,700 and 40,100/15,680. We note:

- The time for the first RRP exchange is significantly longer. The Proxmark traces showed us that for the first challenge only, the card responds to the *ERRD* command by sending a `wait` request and then the response.
- The time taken depends on the distance of the card from the reader, and not so much on the angle. A correlation analysis confirmed this dependence. Similar correlations were seen for the other APDUs that were measured.
- The standard deviations did not vary between positions and was approximately 13,170, 15,700 and 15,680µs for the different nonce exchange rounds.
- The timings measured inside the replay program are significantly longer than those reported here, there is an approximately 17 ms difference (see, for example, Figure 14 in Appendix C which shows the results for all of the first *ERRD* command exchange (denoted as rrp 1) measurements). We initially thought that these delays were due to the time taken to communicate with the ACR122u reader, but subsequent tests with an Adafruit PN532 interface connected to a Raspberry Pi showed that the delays arose due to the PN532's handling of the protocol. (The ACR122u also uses the PN532 IC to handle the NFC signalling).

In conclusion, all measurements, for all the cards tested, showed that timings at Level 3 are far less stable than at Level 1. This suggests that timing for relay-protection should, for better all-round security, be done at Level 1 for EMV (and for other application-layer protocols).

D. Relaying & Disclosure to Mastercard

Given the observed Level 3 timing variation, we looked at the feasibility of relaying the timed nonce exchange from the PayPass-RRP test card. Using the simple setup shown in Fig. 8 we found that it is indeed possible to relay at Level 3 faster than the time taken by the card in the worse position to complete a normal run. While holding the test PayPass-RRP

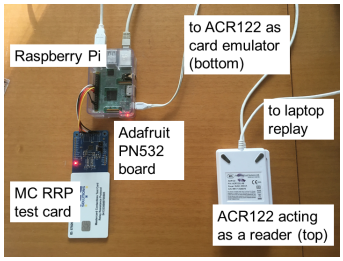

Fig. 8. Relay setup used to investigate possible PayPass-RRP relaying.

card, we presented it to the reader and of ten trials, three gave timings within those seen in our measurements (for rrp 1, these times were 67.79, 74.70 and 77.05ms, while the maximum seen for a direct replay was 79.62ms.).

We discussed this relay attack against the test PayPass-RRP card, and all the measurements, with Mastercard. They welcomed the research and our suggestion that doing the timed nonce-exchange at Level 1 may indeed be more robust against even slightly stronger relay "boxes". They also mentioned that the three trials in the *ERRD* command should not be seen only as an opportunity for re-tries by the relayer, but also as a usability/recoverability feature: correcting human error, and alerting the user that the card is in a non-ideal position. Finally, they mentioned that newer cards (test or commercial), which may appear soon, will be faster than our test card, and so much harder to relay.

VII. A New Level1-based Relay-Protection for EMV: The `L1RP` Protocol

A. Protocol Design

Our `L1RP` protocol is an extension of the Mastercard EMV protocol, and it provides relay resistance by timing a nonce exchange at Level 1. The aim of our protocol is to ensure that the EMV reader cannot successfully complete an EMV transaction unless the card responds to the reader with its nonce within the given time bounds. As in Mastercard's RRP protocol the card also returns timing information to enable the reader to adjust its thresholds to allow for different timing profiles on different cards.

We work in the threat model outlined in Section III. To this we add the requirement that cards must be backwards compatible with readers that do not support relay resistance, without the possibility of downgrade attacks. Additionally, we must ensure that the timing profile from the card can be authenticated by the reader.

Protocol Overview: Following our findings in Section VI, our `L1RP` protocol takes ideas from both the Mastercard and the Visa relay-protections, improves on aspects of them and combines the result into a new protocol.

Our `L1RP` protocol moves the timed nonce-exchange, proposed in Mastercard PayPass-RRP, into the ISO 14443 Level 1 commands. This is informed by our measurements in Section VI-B, where we concluded that Level 1 Relay Resistance makes the RTT-measurements more reliable. As with the proposed Visa protocol, we tie together data at Level 1

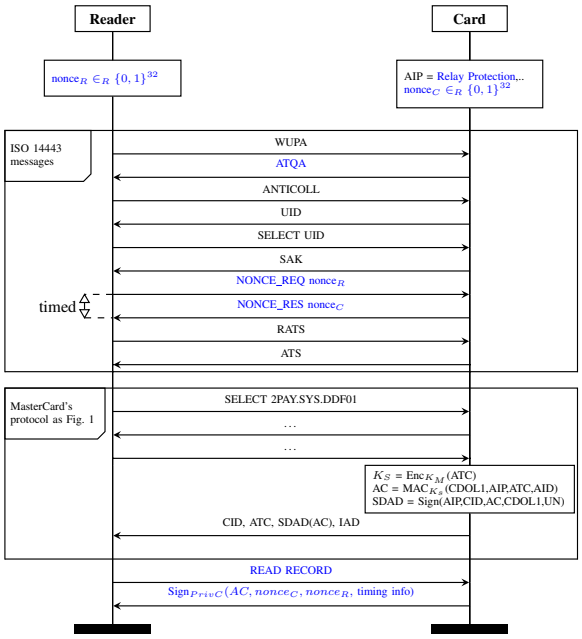

Fig. 9. Our new L1RP protocol

with the EMV application authentication at Level 3. However, note that at Level 1, we include nonces issued not just from the card but from the reader as well, to avoid the problems with the Visa proposal that we highlighted in Section V.

A run of our protocol L1RP is shown in Fig. 9. The card signals to the reader that it supports our protocol using one of bits in the ISO 14443 Answer to Request (ATQA) message reserved for future use (bit 6 & bits 9-16). A bit value of 0 means the card does not support the nonce exchange, whereas a value of 1 will signal that the exchange is supported.

Level 1 Relay Resistance: As per ISO 14443-3 [23] and outlined in Appendix A, after a card responds with a Select Acknowledge (SAK) message, it enters the ACTIVE state. The protocol activation command (RATS) can then be sent by the reader to start the application level protocol. However, a card that is compliant with ISO 14443-3, and in the ACTIVE state can accept *proprietary* commands from the reader instead. This allows us to introduce our new Level 1 command, *NONCE_REQ*, coupled with our *NONCE_RES* response; they are used to execute the nonce exchange at the ISO 14443 level. If the reader and card support our protocol, the reader sends the NONCE_REQ command with a 32-bit nonce. A reader that does not support the protocol will send the RATS command, missing this NONCE_REQ command. The L1RP card replies with the NONCE_RES response, with a 32-bit nonce. The NONCE_REQ – NONCE_RES exchange is *timed* by the reader, to prevent MitM attackers from relaying the messages. See Fig. 10. After this exchange, the RATS command can be used to continue the protocol.

Nonce Generation by the Card: Book 4 of the EMV specification [35] (page 57) suggests that nonces on EMV cards should be generated using specialist algorithms and circuits, e.g., via a PRNG (pseudorandom number generator), but this is left to the card implementer. We opt for this EMV-driven approach, that is – we do not specify our card-side nonce generation at the low-level. Normally, this is proprietary to manufacturers (e.g., NXP), and should adhere to the AIS 20/31 requirements (PTG.1/2/3, DRG.1/2/3/4) [36] for PRNG security certification. Should a manufacturer implement our protocol, we recommend similar PRNG-security practice guidelines, as well as checking developments in the field of PRNGs for cards [37] and for other computationally-limited devices [38].

Including Proof of the Level 1 Nonce Exchange in Level 3: At the EMV application level our protocol runs exactly as the Mastercard PayPass protocol, but after receiving the AC, the L1RP-compatible reader will use a new READ RECORD command to read out the timing information for the card (expected, maximum and minimum response times) and the Level 1 nonces signed by the card along with the AC. These signed nonces attest that the Level 1 nonce exchange was done with the the card, and binds the nonces to the AC.

Protection from Downgrade Attacks: The card's ability to run our protocol is recorded in the AIP, described in Section II-A, it is signed by the card in the SDAD which is checked by the reader. So, any attempts to downgrade a card by changing the AIP, to make the reader believe it doesn't support L1RP will be detected by the reader as a mismatch in the SDAD.

Separate Relay Resistance and Payment Proofs: We note the separation of relay resistance and payment proofs (e.g., adding a "special" SDAD-like message sent by the card for the distance bounding proofs). It would have been possible to put the Level 1 nonces into the SDAD, rather than having a separate new signature. We did not do this because: (1) we did not want the possibility that a normal EMV transcript could be confused with a transcript of our protocol; (2) this separation allows for better backwards-compatibility especially over various operation modes; (3) the separation allows clearer security proofs (adhering to both established EMV as well as distance bounding security models).

L1RP vs Visa's Level 1 Relay-Protection: We now summarise why L1RP does not suffer from the attack we showed in Section V against Visa's Level 1 relay-protection protocol. Recall that the security claims in Visa's relay-protection protocol rely on a value called UID coming only from the card/phone's side, i.e., the claimed security does not hinge on any input from the reader side.

Unlike the Visa protocol, our L1RP protocol uses random nonces (as opposed to UIDs) and they are issued from *both* the card and the reader's sides of the EMV protocol. Even if we weakened our protocol, and let the card/phone send a UID as per Visa's case, just applying our attack from Section V *as is* would still not work against this weakened L1RP. In our attack, only the card's UID is manipulated maliciously; if this were applied to L1RP, the victim's phone would not receive the nonce of the real reader and the EMV-level checks in the last step of L1RP would fail.

Now assume we lifted our attack in Section V to a MitM who was resetting both the card and reader's nonces in L1RP.

In L1RP, the reader's nonce is sent out first, and the nonce exchange is timed by the reader. So, to fall within the allowed time-bound, the attacker would have to guess preemptively the nonce of the reader such as to send it in time to the victim's phone. This would happen with a probability of 2^{-32}, under our assumption of pseudorandom nonces.

In summary, the security of our protocol is based on: (1) session-identifying, pseudorandom data coming not just from one side but from both sides of a two-party protocol; (2) timing this exchange of the random data, so not only can the MitM attacker not guess it, given its pseudo-random nature, but this MitM cannot relay it effectively either.

B. Formal Verification of Security

We extend our model of Mastercard's contactless protocol from Section IV-G with support for our L1RP protocol. New rules model the sending of the Level 1 ATQA message that will indicate if a device supports our protocol and, for devices that do, the NONCE_REQ, NONCE_RES nonce exchange messages, as well as the reader asking for the DB proof. We also enhance the model with the reader checkiing of this DB-proof, the protocol-type recorded in the AIP, and add lemmas to verify the lack of downgrade attacks.

Our model includes devices and readers that do and do not support our L1RP protocol. We check that any combination of devices and readers can finish an EMV transaction. We check protection from downgrade attacks by ensuring that a reader that supports our L1RP protocol can only ever finish a run, without DB protection, if the device it is running the protocol with does not support our L1RP protocol.

To verify protection from relay attacks we use the framework of Mauw *et. al.* [5], who present a definition of *causality-based secure distance-bounding*. This can be used to verify the correctness of distance-bounding protocols without the need to explicitly model time and locations. This model assumes a protocol with a verifier (the terminal) and a prover (the device). The verifier will have a timed phase; in our Tamarin model we indicate this by a DB_Start action when the terminal sends the NONCE_REQ message and a DB_End action when the terminal receives the NONCE_RES message.

The action that the device/prover is expected to perform during this phase (the reply to the NONCE_REQ message) is tagged with a DB_Action label, and we add a DB_Claim label to the terminal Tamarin rule that verifies the DB proof. All of these actions are parameterised on the identities of the prover and the verifier, and the nonces used in the transaction.

Mauw *et. al.* [5] show that the standard notion of distance-bounding security, regarding MitM attackers, holds if whenever there exists a DB_Claim(P,V,Nc,Nr) action in a trace then there must have been a preceding DB_End (V,Nc,Nr) action, preceded by a DB_Action (P,Nc,Nr) preceded by a DB_Start(V,Nr). I.e., the prover must have acted during the timed phase of the verifier or, P or V must have been compromised. This definition a la Mauw *et. al.* [5] is easy to check in Tamarin, which takes 327 seconds to show that it holds and

```
[usb] pm3 --> hf 14a list
[+] Recorded activity (trace len = 137 bytes)
[=] Start = Start of Start Bit, End = End of last modulation.
Src = Source of Transfer
[=] ISO14443A - All times are in carrier periods (1/13.56MHz)

  Start |    End | Src | Data (! denotes parity error) | CRC | Annotation
--------+--------+-----+-------------------------------+-----+------------
      0 |    992 | Rdr |52                             |     | WUPA
   2100 |   4532 | Tag |04 01          ← 01 indicates to do the random nonce exchange
   7040 |   9504 | Rdr |93 20                          |     | ANTICOLL
  10548 |  16372 | Tag |88 f6 53 90 bd                 |     |
  19072 |  29536 | Rdr |93 70 88 f6 53 90 bd ec 2e |  ok | SELECT_UID
  30644 |  34228 | Tag |20 fc 70                       |     |      select completed
  36480 |  44704 | Rdr |99 76 0a 4f 8a 7e e3           |     | NONCE_REQ
  45748 |  51572 | Tag |c0 7d 6a d1 06          timed  |     | NONCE_RES
  56192 |  60960 | Rdr |e0 80 31 73                    |  ok | RATS
  62004 |  69044 | Tag |04 58 80 02 13 ce              |  ok |
```

Fig. 10. The nonce exchange proposal.

that our protocol provides the distance bounding protection we require. The full annotated model is in L1RP.spthy in [9].

C. Practical Experiments

The proposed extension to the current Level 1 protocol was tested using two Proxmarks. On one Proxmark, we implemented the reader protocol including, if required, the nonce-exchange, and on the other, we implemented a simulation of the card, again including our proposed nonce-exchange. On top of this L1RP-driven implementation, our Proxmark-based programs include code to measure the message times needed to assess our implementation. The timings (and other tests) only refer to Level 1 commands up to and including the RATS command, as these are the ones used in L1RP.

The results from one of our timing tests are shown in Fig. 10, which also highlights the proposed changes to the protocol (in this case, for a 4 byte UID).

Proxmark Implementation of L1RP: We modified the Proxmark code to add two Level 1 functions, one to implement the reader side and the other the card side. These were:

- `noncerdr` this implements the reader side and executes the protocol up to the RATS message. If bit 9 of the ATQA message is set it carries out the nonce exchange. The nonce is obtained using the Linux `getrandom` function which gets random bytes from /dev/random.
- `noncesim` this simulates a card which expects to do a nonce exchange and so sets *bit 9* in ATQA. The card is given a random UID and nonce, these are both obtained using the Linux `getrandom` function.

Notes on the implementation:

- We used ATQA *bit 9* to flag that the card can do the nonce exchange.
- To identify the reader's nonce exchange message, we chose an unused message code, `0x99`. On the card side, we send the nonce and a Block Check Character (BCC), as is done when a card sends its UID.
- We tested the backwards compatibility of our new card and reader implementations by verifying that they would work with an existing EMV reader and EMV card.

Time Measurements for L1RP: The RTT using the Proxmarks was measured in 30 tests with a mean of 1,116µs with a standard deviation of 2.3µs. This time is very similar to the time measured for the SELECT UID command for

the Proxmarks and for 'actual' payment cards (see Table I, Appendix C), which suggests time bounding this exchange may work well. These Proxmark results are encouraging, and we will implement `L1RP` and obtain more measurements.

Challenges when Relaying over `L1RP`: Analysing our Proxmark-issued timing data, we observed that most of the time for the nonce-exchange is taken sending and receiving the data (≈1037µs), i.e., not preparing the data. This makes a relay with unmodified COTS hardware difficult.

Reading the data directly and using this to modulate a carrier signal used to transmit the data to a receiver where it is demodulated and used to drive a loop antenna could provide a low-level relay, but this is a much more difficult undertaking than the threat model we envisage (see Section III). This type of lower-level relay has been illustrated for an LF RFID system by Francillon et al. [19] who used it to attack a car's RFID key-entry system. However, this was for a LF RFID system and not the HF RFID system used for EMV payment cards. While implementing such a low-level (Level 1) attack on HF RFID systems is theoretically feasible it has yet to be demonstrated.

VIII. Related Work

Practical (relay) attacks against EMV were demonstrated in a range of works, e.g. [39], [26]. Other attacks have been reported against EMV. Two over the limit attacks, [2], [7] for Visa are discussed in sub-section II-B. In [8], a Mastercard card is made to look like a Visa card in order to perform an over the limit attack on Mastercard, which is now patched. [2] describes an attack that makes an offline reader accept an invalid AC for Visa cards (an attack that was also described in [26, p.5]. Murdoch *et. al.* [40] presented a MitM attack against contact based EMV, and evidence in practice is presented in [11]. Emms *et. al.* [41] show how the non-TAP & PIN card limit can be bypassed by switching currencies.

None of these attacks would work against EMV on mobile devices if CDCVM authentication was always required, and these attacks require a relay, and so would be stopped by a protocol that prevented relaying (the main focus of our work). We are the first to mount attacks bypassing authentication to make illicit payments in *mobile* payment apps, and the first to look at and exploit their different operating modes.

Distance Bounding: DB protocols were introduced by Brands and Chaum [42] with the view of combating relay-based MitM attacks. In distance-bounding protocols a reader and a tag (RFID card, smart card, etc.) run a challenge-response protocol in which the reader measures the RTT and if this is smaller than a given bound, the reader assumes physical proximity between it and the card. The survey in [43] gives numerous distance-bounding protocols, security concepts and attacks in distance bounding.

RRP closely follows the previously proposed PaySafe protocol [26]. RRP was first formally verified in [5]. [44] performs a timing analysis of RRP and finds it secure, but without varying the positions of the card in the field, as we do.

NXP provides a distance-bounding protocol on their DESFire cards. Patents [45], [46] are the only public source of information on this, first recounted in [47]. These show that NXP's distance bounding uses a Level 1 nonce exchange, however it provides no details of how/whether this distance bounding is tied to the upper level protocols or applications.

Symbolic Verification of EMV: Past symbolic-verification models of EMV include [48], [2], [8] and of EMV with distance bounding [47], [26], [5], [49], [44].

The most up-to-date and complete formal model for EMV is in [2]. We follow this model (for contactless Visa, Mastercard) and extend it to encode mobile apps in transport and non-transport models, in a unitary manner (CDA, DDA, high/low in one model). We add a full model of the IAD to their models, and show that Visa/bank checking the IAD stops their attack.

Symbolic Verification of Distance Bounding: Automatable symbolic verification of distance-bounding protocols was first proposed by Basin *et. al.* [50]. Many recent developments exist. Nigam *et. al.* [51] develop a distance bounding checking tool based on Maude. Rowe et al, [52] present a framework based on strand spaces. Debant *et. al.* in [49], [53] are the only symbolic models to encode time and location explicitly. Symbolic models tend to eliminate timing even if this is considered in some of the theoretical models. Chothia *et. al.*, suggest a framework without an explicit model of time [26], [47]. Mauw *et. al.* [5], [54] have a proof that, under certain conditions and for certain protocols (e.g., with one DB exchange, with no other timing notions, without certain message inter-dependencies), timed DB-security can be reduced to checking an order of events on traces; this elegant definition can be easily checked in the Tamarin prover and is the method we use. Boureanu, *et. al.* [44] extends this to support mobility.

IX. Conclusions

We investigated mobile payments-apps in different operation modes, showing their different defences against bypassing authentication in transport mode. This allows us to make fraudulent Visa payments with locked iPhones of any value we wish. This vulnerability is due to the lack of checks performed on the iPhone *combined* with the lack of checks at the Visa back end. Apple Pay with Mastercard is not vulnerable and nor are Mastercard and Visa with Samsung Pay.

We have looked at the practical security of Mastercard and Visa's relay-counteraction solutions for EMV, and found that the former could potentially be functionally improved, whereas the latter is lacking in security. To fix these issues, we have proposed an EMV relay-resistant protocol that combines ideas from both the Mastercard and Visa relay-countermeasures proposals. We have proved our protocol formally correct and described a trial implementation of its distance-bounding part, on Proxmarks; this shows our protocol is practical for EMV cards, and also for use with Apple Pay and Samsung Pay to stop relaying to the so-called express transit systems.

Acknowledgements: This work is part of the "TimeTrust" project, funded the UK's National Cyber Security Centre (NCSC). We thank Mastercard UK and Visa Research for providing useful insights and feedback.

REFERENCES

[1] EMVCo, "Book C-2 kernel 2 specification v2.7. EMV contactless specifications for payment system," Feb 2018.

[2] D. A. Basin, R. Sasse, and J. Toro-Pozo, "The EMV standard: Break, fix, verify," in *Security and Privacy (SP)*, 2021.

[3] C. Yuexi, M. Kekicheff, M. Top, and H. Ngo, "Binding Cryptogram with Protocol Characteristics," 2019, uS Patent App. 16/348,085.

[4] H. Shan and J. Yuan, "Man in the NFC," DefCon 25, 2017.

[5] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua, "Distance-bounding protocols: Verification without time and location," in *IEEE Symposium on Security and Privacy, SP*, 2018.

[6] "Lab 401 proxmark 3 rdv 4." [Online]. Available: https://lab401.com/products/proxmark-3-rdv4

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

[8] D. Basin, R. Sasse, and J. 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.

[9] "Practical EMV relay protection: Artefacts." [Online]. Available: https://gitlab.com/practical_emv

[10] "Visa mobile contactless payment specification," https://technologypartner.visa.com/Library/Specifications.aspx.

[11] H. Ferradi, R. Geacuteraud, D. Naccache, and A. Tria, "When organized crime applies academic results - a forensic analysis of an in-card listening device," *IACR Cryptology ePrint Archive*, vol. 2015, p. 963, 2015. [Online]. Available: https://eprint.iacr.org/2015/963

[12] Visa, "Visa merchant data standards manual," 2019. [Online]. Available: https://usa.visa.com/content/dam/VCOM/download/merchants/visa-merchant-data-standards-manual.pdf

[13] EMVCo, "Book C-3 kernel 3 specification v2.6. EMV contactless specifications for payment system," Feb 2016.

[14] ——, "EMV payment tokenisation specification – technical framework v2.2," 2020.

[15] "Visa contactlesspayment specification," https://technologypartner.visa.com/Library/Specifications.aspx.

[16] EMVCo, "Book 2: Security and key management," Nov 2011.

[17] Visa Research, "Visa proposal for level 1 protocol parameter binding against relay attack, version 0.1," 2017.

[18] 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.

[19] A. Francillon, B. Danev, and S. Capkun, "Relay attacks on passive keyless entry and start systems in modern cars," Cryptology ePrint Archive, Report 2010/332, 2010, https://eprint.iacr.org/2010/332.

[20] M. Singh, P. Leu, and S. Capkun, "UWB with pulse reordering: Securing ranging against relay and physical-layer attacks," in *Network and Distributed System Security Symposium, NDSS*, 2019.

[21] "Where you can travel on public transport with apple pay," 2020. [Online]. Available: https://support.apple.com/en-gb/HT207958

[22] "Samsung faq," 2020. [Online]. Available: https://www.samsung.com/uk/samsung-pay/faq/

[23] ISO, "14443-3: 2018 – Identification cards – Contactless integrated circuit cards – Proximity cards – Part 3: Initialization and anticollision," International Organization for Standardization, Standard, 2018.

[24] "Common vulnerability scoring system version 3.1: Specification document," 2019. [Online]. Available: https://www.first.org/cvss/specification-document

[25] "Common vulnerability scoring system version 3.1 calculator," 2019. [Online]. Available: https://www.first.org/cvss/calculator/3.1

[26] T. Chothia, F. D. Garcia, J. de Ruiter, J. van den Breekel, and M. Thompson, "Relay cost bounding for contactless EMV payments," in *Financial Cryptography (FC)*, ser. LNCS, 2015.

[27] Citi bank, "Merchant category codes," 2015. [Online]. Available: https://www.citibank.com/tts/solutions/commercial-cards/assets/docs/govt/Merchant-Category-Codes.pdf

[28] EMVCo, "Book 3: Application specification," Nov 2011.

[29] B. Blanchet, "Security protocol verification: Symbolic and computational models," in *Principles of Security and Trust*, 2012.

[30] D. Dolev and A. Yao, "On the Security of Public-Key Protocols," *IEEE Transactionson Information Theory 29*, vol. 29, no. 2, 1983.

[31] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov, "Undecidability of Bounded Security Protocols," in *Workshop on Formal Methods and Security Protocols (FMSP'99)*, 1999.

[32] S. Jasek, "A 2018 practical guide to hacking nfc/rfid," 2018. [Online]. Available: https://smartlockpicking.com/slides/Confidence_A_2018_Practical_Guide_To_Hacking_RFID_NFC.pdf

[33] Android Developers, "Host-based card emulation overview," 2019. [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc/hce

[34] EMV, "Contactless specifications for payment systems, book A, version 2.6 – architecture and general requirements," 2016.

[35] EMVCo, "Book 4: Cardholder, attendant, and acquirer interface requirements," May 2017.

[36] W. Schindler and W. Killmann, "Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications," in *Workshop on Cryptographic Hardware and Embedded Systems*, 2002.

[37] R. N. Akram, K. Markantonakis, and K. Mayes, "Pseudorandom Number Generation in Smart Cards: An Implementation, Performance and Randomness Analysis," in *New Technologies, Mobility and Security*, 2012.

[38] A. Francillon and C. Castelluccia, "TinyRNG: A Cryptographic Random Number Generator for Wireless Sensors Network Nodes," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops*, 2007.

[39] L. Francis, G. Hancke, and K. Mayes, "A practical generic relay attack on contactless transactions by using NFC mobile phones," *International Journal of RFID Security and Cryprography (IJRFIDSC)*, 2013.

[40] S. J. Murdoch, S. Drimer, R. J. Anderson, and M. Bond, "Chip and pin is broken," in *Symposium on Security and Privacy, S&P*, 2010.

[41] M. Emms, B. Arief, L. Freitas, J. Hannon, and A. van Moorsel, "Harvesting high value foreign currency transactions from EMV contactless credit cards without the PIN," in *Computer and Communications Security, CCS*, 2014.

[42] S. Brands and D. Chaum, "Distance-bounding protocols," in *Advances in Cryptology – EUROCRYPT*. Springer, 1993, pp. 344–359.

[43] G. Avoine, M. Bingol, I. Boureanu, S. Capkun, G. Hancke, S. Kardas, C. Kim, C. Lauradoux, B. Martin, J. Munilla, and et al, "Security of Distance-Bounding: A Survey," *CSUR*, vol. 4, 2017.

[44] I. Boureanu, T. Chothia, A. Debant, and S. Delaune, "Security Analysis and Implementation of Relay-Resistant Contactless Payments," in *Computer and Communications Security (CCS)*, 2020.

[45] P. Thueringer, H. De Jong, B. Murray, H. Neumann, P. Hubmer, and S. Stern, "Decoupling of measuring the response time of a transponder and its authentication," November 2008.

[46] P. Janssens, "Proximity check for communication devices," April 2015.

[47] T. Chothia, J. de Ruiter, and B. Smyth, "Modelling and analysis of a hierarchy of distance bounding attacks," in *USENIX Security*, 2018.

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

[49] A. Debant, S. Delaune, and C. Wiedling, "Proving physical proximity using symbolic models," Univ Rennes, CNRS, IRISA, France, Research Report, Feb. 2018.

[50] D. Basin, S. Capkun, P. Schaller, and B. Schmidt, "Formal reasoning about physical properties of security protocols," *Transactions on Information and System Security (TISSEC)*, 2011.

[51] V. Nigam, C. Talcott, and A. Aires Urquiza, "Towards the automated verification of cyber-physical security protocols: Bounding the number of timed intruders," in *European Symposium on Research in Computer Security (ESORICS)*, 2016.

[52] P. D. Rowe, J. D. Guttman, and J. D. Ramsdell, "Assumption-Based Analysis of Distance-Bounding Protocols with CPSA," in *Logic, Language, and Security*, 2020.

[53] A. Debant, S. Delaune, and C. Wiedling, "Symbolic analysis of terrorist fraud resistance," in *European Symposium on Research in Computer Security, (ESORICS)*. Springer, 2019.

[54] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua, "Post-collusion security and distance bounding," in *Computer and Communications Security, CCS*, 2019.

[55] ISO, "14443-4: 2018 – Identification cards – Contactless integrated circuit cards – Proximity cards – Part 4: Transmission protocol," International Organization for Standardization, Standard, 2018.

[56] ——, "7816-4: 2020 – Identification cards – Integrated circuit cards Part 4: Organization, security and commands for interchange," International Organization for Standardization, Standard, 2020.

[57] EMVCo, "EMV Contactless Specifications for Payment Systems; Book D; EMV Contactless Communication Protocol Specification," 2016.

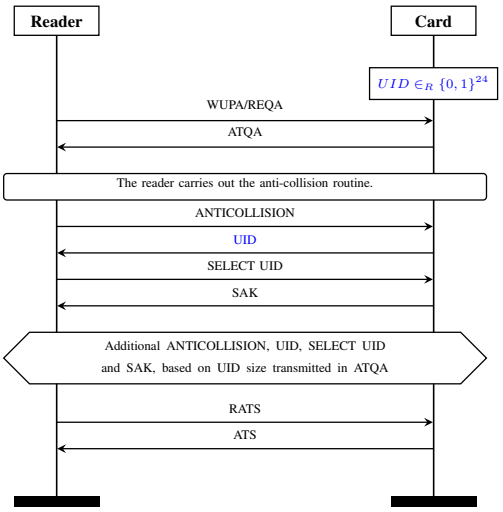

Fig. 11. ISO 14443 Protocol Sketch (when there is a single card in the field)

APPENDIX A
THE ISO 14443 PROTOCOL

There are several standards for proximity cards defining their electrical characteristics and how the fields are modulated and data transmitted between a reader and a proximity (RFID) card. Contactless payment cards use the ISO 14443 standard [23], [55] for the lower levels of the communication stack. There are four documents in this standard, the first two define the electrical characteristics and modulation schemes to be used. We do not need to consider these in the work that we are reporting. We focus on ISO 14443-3, which defines how a card establishes a link with a reader and on ISO 14443-4, which gives the transmission protocol to be used after the card is "linked" to the reader. Built on top of this is a protocol, ISO 7816-4, which defines how data is packaged up into APDUs. The EMV specifications refer to the 14443 standards as Level 1, the ISO 7816-4 standard [56] as Level 2, and the EMV messages themselves as Level 3.

A. ISO 14443-3

The ISO 14443-3 standard [23] defines how the initial link between a reader and a card is setup (see Fig. 11). The reader is regularly polling for proximity cards by sending WUPA or REQA messages. When a card enters the magnetic field generated by the reader, the card draws energy from the reader and when it receives a WUPA or a REQA message it starts the protocol by replying with an ATQA message.

The ATQA response gives the reader information about the card that it can then use in the steps that follow. In particular, it tells the reader the size of the UID that the card uses and this is used in the next anti-collision phase. Several cards could respond to the same WUPA/REQA message and the anti-collision phase is used to select a single card for the next stage. During this process a number of ANTICOLLISION and SELECT messages from the reader are used, finally selecting a single card. In the final stage the reader sends a SELECT message with the UID being selected and the card responds with an acknowledgement (SAK). This acknowledgement tells the reader that the card is compliant with ISO 14443-4 and can send the RATS message (bit 3=0 and bit 6=1). Fig. 11 shows the process when there is only a single response and the anti-collision phase is cut short. This applies in the tests that we have been doing and it should also be noted that, for payment cards, if the reader detects a collision the payment is rejected [57]. Once a card is selected the reader moves on to the next part of the protocol with the card in the ACTIVE state.

B. ISO 14443-4

When a card is selected and in the ACTIVE state, the ISO 14443-4 standard [55] specifies how the communication should proceed. It starts with the reader sending a RATS command and the card replying with an Answer to Select (ATS) response. These two message setup parameters for the communications that follow, such as limits on the size of frames that can be sent, or received, and for the card, timing parameters and the "historical bytes". Historical bytes are optional and can be used to transfer card specific information from the card to the reader. This standard also details how large frames should be split, sent, and then re-combined, and how the card can request more time to respond by sending frame waiting time extension requests. For our reader all of these operations are carried out by the NFC chip, it sets the parameters for the transfers (except the historical bytes) and once the RATS–ATS exchange is completed, frames to and from the layer above are handled transparently. Splitting frames will clearly affect the timing, so any timed exchange should use as small a frame as possible.

APPENDIX B
THE PAYWAVE & VISA-L1 PROTOCOLS

The VISA-L1 protocol is an extension of the Visa PayWave protocol shown in Fig. 2.

The extension of the Visa PayWave protocol to the VISA-L1 protocol is one where the UID (the blue message in Fig. 11) is randomised (as opposed to being static) and this UID is included in the SDAD. The Visa specification document [17] only states that the UID should be *cryptographically bound* with the Level 3 protocol session data. Doing this by including the UID specifically inside the SDAD (as opposed to other parts of the protocol, such as the AC) is based on our conversations with Visa Research.

Fig. 12 shows the diagram of the relay attack against VISA-L1, which can be done with a rooted Android phone.

APPENDIX C
TIMINGS RESULTS

For the Mastercard-RRP test card, in addition to measuring the RRP messages, we also measured the response times for

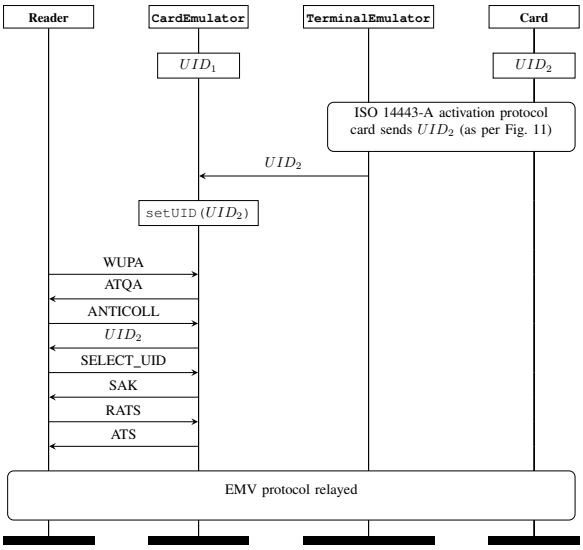

Fig. 12. Relay of card UID against Visa's relay protection protocol

the other messages used. For comparison, we also measured the response times for the messages from a "normal" Mastercard and Visa card running their protocols. Apart from the test card, none of our Mastercards responded to the RRP messages.

The timings for the Level 1 messages are given in Table I. We see that the average response times for the WUPA, ANTICOLL and SELECT UID messages are all very similar. Although there are some differences in the variability, note that these figures are in µs. For all three of these messages there is no correlation between the time taken and the distance of the card from the reader (see Table III). For the RATS messages the two Mastercards have similar timings, while for the Visa card the response time is shorter, but so is the message size. However, the difference in length does not totally explain the differences. The timings for the responses to the RATS message are correlated with the distance from the reader. It is difficult to be certain, but it seems that for the first three of these messages the response is automatic, once energised by the field the card can immediately respond. The RATS message clearly needs some processing to be done before it can respond and the time for this depends on the strength of the energising electromagnetic field. This is also seen in the Level 3 results given below. To minimise time variations, any nonce exchange at this level should be organised so that any processing required is kept to a minimum – the card's nonce should be generated as soon as the card enters the field and before it responds to the WUPA command.

We now present selected timing results for Level 3 commands, we consider the GPO message with its reply, as well as the nonce exchanges of the Mastercard-RRP card. The results obtained are summarised in Table II. For the GPO 1 message, plots of the message timings with respect to card distance and

TABLE I
LEVEL 1 REPLY TIME MEASUREMENTS FOR MASTERCARD AND VISA CARDS, IN µS.

command	avg	std	bytes	range
Mastercard-RRP				
wupa	340.46	0.91	2	2.36
anticoll	699.15	0.88	5	2.36
select uid	1123.91	0.89	3	2.36
rats	2875.93	163.82	22	549.85
Mastercard				
wupa	340.09	6.17	2	146.31
anticoll	699.08	0.70	5	2.36
select uid	1127.77	12.38	3	208.85
rats	3231.61	286.50	21	989.97
Visa				
wupa	339.34	13.55	2	147.49
anticoll	699.29	0.76	5	2.36
select uid	1127.29	17.58	3	208.85
rats	1799.91	82.73	12	246.61

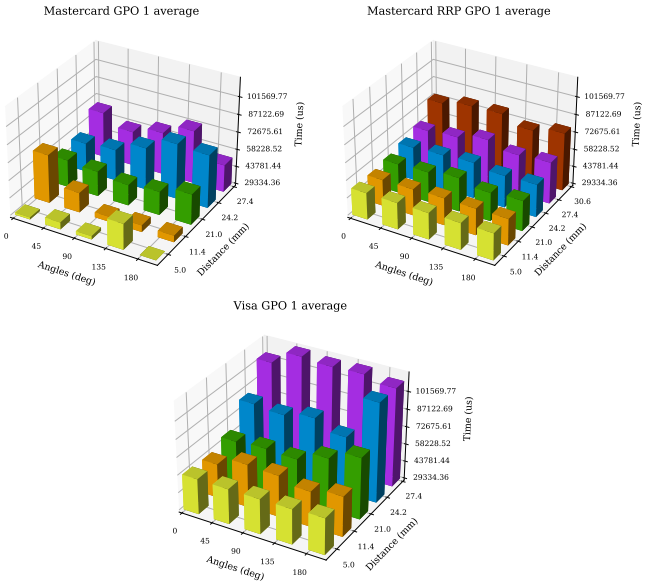

Fig. 13. Average reply times for the EMV GPO command and response

angle are shown in Fig. 13. Just as for the RATS command, there is a clear correlation between the response times and the distance of the card from the reader for all of these commands (see Table III).

We also carried out *correlation analysis* on the timing measurements, with respect to the different distances we recorded. We tested the hypothesis of whether there is a linear relationship between the distance between a card and a reader, and the reply times observed, for each Level 1 and Level 3 command. The data reveals that for most Level 1 commands (WUPA, ANTICOLLISION and SELECT UID), there is no significant relationship. However, for Level 3 commands there is a strong significant correlation between the two variables. This confirms the intuitive observations formed from the average and standard deviation analysis, that performing distance bounding at Level 1 would allow tighter

TABLE II
LEVEL 3 REPLY TIME MEASUREMENTS FOR MASTERCARD AND VISA CARDS ACROSS ALL ANGLES AND DISTANCES, IN µS.

command	avg	std	bytes	range
Mastercard-RRP				
Select 2Pay	97371.78	14495.52	42	76842.48
Select AID	67223.86	6617.87	112	23995.28
GPO 1	58862.01	8926.51	51	28307.96
rrp 1	48429.66	6373.18	17	20200.59
rrp 2	34920.51	8672.56	17	28883.78
rrp 3	34919.93	8675.94	17	27638.94
read record: 01, SFI: 2	86088.06	5675.82	178	17990.56
read record: 01, SFI: 3	104328.99	5765.41	203	18273.75
read record: 01, SFI: 4	41900.57	5419.67	40	17139.82
read record: 02, SFI: 5	83108.45	5641.58	144	17851.33
checksum	23401.71	8088.24	5	25137.46
Mastercard				
Select 2Pay	39790.21	4472.72	81	15263.72
Select AID	45680.31	7105.98	87	25137.46
GPO 1	51490.57	14234.50	25	48857.82
GPO 2	390503.14	129159.73	174	571411.21
read record: 01, SFI: 2	63134.47	6804.63	180	23153.98
read record: 01, SFI: 4	85438.76	7171.63	201	44305.60
read record: 02, SFI: 4	85050.24	7179.88	241	26033.04
Visa				
Select 2Pay	37100.56	4485.14	81	13015.93
Select AID	37755.15	6217.96	91	17715.63
GPO 1	100902.32	21548.54	81	107037.17

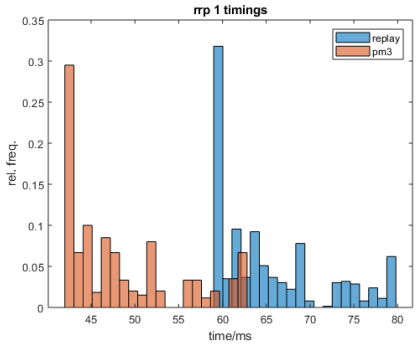

Fig. 14. RRP 1 timings obtained from the Proxmark and from inside the replay program.

time bounds to be imposed.

TABLE III
CORRELATION COEFFICIENT AND PROBABILITIES FOR REPLY TIME MEASUREMENTS FOR VISA AND MASTERCARD CARDS.

	Corr coef	p-value
Mastercard-RRP		
wupa	-0.08	0.06
anticoll	-0.03	0.47
select uid	-0.03	0.41
rats	0.82	< 0.05
Select 2Pay	0.80	< 0.05
Select AID	0.83	< 0.05
GPO 1	0.80	< 0.05
rrp 1	0.82	< 0.05
rrp 2	0.85	< 0.05
rrp 3	0.85	< 0.05
read record: 01, SFI: 2	0.82	< 0.05
read record: 01, SFI: 3	0.82	< 0.05
read record: 01, SFI: 4	0.82	< 0.05
read record: 02, SFI: 5	0.82	< 0.05
checksum	0.82	< 0.05
restore	0.88	< 0.05
Mastercard		
wupa	0.02	0.62
anticoll	-0.16	< 0.05
select uid	0.06	0.14
rats	0.74	< 0.05
Select 2Pay	0.72	< 0.05
Select AID	0.72	< 0.05
GPO 1	0.74	< 0.05
GPO 2	0.73	< 0.05
read record: 01, SFI: 2	0.73	< 0.05
read record: 01, SFI: 4	0.72	< 0.05
read record: 02, SFI: 4	0.73	< 0.05
Visa		
wupa	-0.01	0.70
anticoll	-0.28	< 0.05
select uid	-0.06	0.12
rats	0.69	< 0.05
Select 2Pay	0.81	< 0.05
Select AID	0.82	< 0.05
GPO 1	0.81	< 0.05
restore	0.78	< 0.05

APPENDIX D
EMV ACRONYMS & MODELS DIFFERENCES

TABLE IV
EMV ACRONYMS USED IN THIS PAPER.

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 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.
CARD	Card Authentication Related Data	Authentication data that may be returned as part of the transaction data. If returned it includes a card nonce and the authorised payment amount.
CDA	Combined DDA and AC	This is one method used for offline card authentication and the signed data and application cryptogram are generated together.
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.
CVM	Cardholder Verification Method	This is used to verify that it is the legitimate cardholder that is presenting the card for payment. There are different verification methods and they have different payment limits applied to them.
CVR	Card Verification Results	Data returned to the Issuer as part of the IAD.
DDA	Dynamic Data Authentication	Used to authenticated the card, but unlike SDA the card is required to sign a nonce from the terminal. This is more secure and is designed to stop replay attacks.
ERRD	Exchange Relay Resistance Data	The EMV command used by the Mastercard protocol for providing relay resistance for contactless cards.
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 [56]).
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	A standard code (ISO 18245) used for retail financial transactions to classify the business type.
MNL	Merchant Name and Location	Just what it says, although it is not always correctly initialised. For TfL it contains the station name.
ODA	Offline Data Authentication	Mode of authenticating the card when the terminal is offline. This may be done using DDA, or SDA.
PAR	Payment Account Reference	A non-financial reference assigned to each unique Payment Account Number (PAN) and used to link a Payment Account represented by that PAN to an affiliated Payment Token.
PDOL	Processing options Data Object List	A list of data sent to the card for it to use when processing the transaction.
SDA	Static Data Authentication	Allows the card to be authenticated when in offline mode using fixed data.
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 may generate and use nonces.

TABLE V
EMV DATA DIFFERENCES BETWEEN BASIN ET. AL. MODELS AND MODELS PRESENTED IN THIS PAPER.

EMV data	Basin et. al. [2]	Our models	Comments
IAD	Format 0/1/3	Format 4	IAD format is given by the IAD length (1st byte) and, for Format 2 & 4, the left nibble of the CVN (2nd byte); cross-checking IAD with transaction amount would prevent limit-bypass attack presented in [2].
CVR (part of IAD)	✗	✓	Format 4 contains a *CDCVM* bit; checking this would prevent limit-bypass attack presented in [2].
MCC	✗	✓	Sent to payment network, even if not present in transaction APDUs; cross-checking MCC with IAD would prevent limit-bypass attacks.