Making Contactless EMV Robust Against Rogue Readers Colluding With Relay Attackers

Tom Chothia[1], Ioana Boureanu[2] **, and Liqun Chen[2]

[1] School of Computer Science, University of Birmingham, UK
[2] Department of Computer Science, University of Surrey, UK

Abstract. It is possible to relay signals between a contactless EMV card and a shop's EMV reader and so make a fraudulent payment without the card-owner's knowledge. Existing countermeasures rely on *proximity checking*: the reader will measure round trip times in message-exchanges, and will reject replies that take longer than expected (which suggests they have been relayed). However, it is the reader that would receive the illicit payment from any relayed transaction, so a rogue reader has little incentive to enforce the required checks. Furthermore, cases of malware targeting point-of-sales systems are common. We propose three novel proximity-checking protocols that use a trusted platform module (TPM) to ensure that the reader performs the time-measurements correctly. After running one of our proposed protocols, the bank can be sure that the card and reader were in close proximity, even if the reader tries to subvert the protocol. Our first protocol makes changes to the cards and readers, our second modifies the readers and the banking backend, and our third allows the detection of relay attacks, after they have happened, with only changes to the readers.

1 Introduction

Wireless and particularly contactless systems, such as the EMV (Europay, Mastercard and Visa) contactless-payment protocols, are vulnerable to *relay attacks*. That is, an adversary can stand near a victim (e.g., a bankcard) and relay signals from that device to a second attacker found near the authentication-verifying party (e.g., a payment terminal). This type of attack has already been used to steal cars[3]. As relayed messages take longer to travel then direct messages, *proximity-checking* or *distance-bounding* (DB) protocols [1] measure the round trip time (RTT) it takes for some authenticating party, called *prover*, to answer challenges sent by an authentication-verifying party, called *verifier*. If the RTT is within a given bound, then there is a low likelihood that a relay attack occurred. As such, the contactless version of the EMV protocol has recently been enhanced with such a relay-counteraction mechanism [5] (see Figure 2), in the style of a previously proposed DB protocol [4]. As with other DB protocols, these newly proposed EMV protocols assume that the reader is honest. However, this threat-model conflicts with the setting of EMV. I.e, in the current EMV protocols, the entity tasked with enforcing the proximity checks is also the one that stands to benefit if these checks are ignored: an EMV reader has an incentive to be dishonest as it will receive the payments from any (relayed) transaction. EMV readers have also been the target of malware (see e.g. [7]), which could also override the RTT-measuring software.

** Tom Chothia and Ioana Boureanu have contributed equally to this work.

[3] See e.g. `http://www.bbc.com/news/av/uk-42132804/relay-crime-theft-caught-on-camera`

	Prevents collusive relay attacks	Provides audit-able evidence	Reader may be offline	Changes to card	Changes to EMV reader	Changes to bank system backend	Checks carried out by
Protocol `PayCCR`	✓	✓	✓	Yes	Yes	No	card
Protocol `PayBCR`	✓	✓	×	No	Yes	Yes	bank
Protocol `PayBCRv2`	×	✓	✓	No	Yes	No	auditor

Fig. 1: A summary of the protocols presented in this paper

The above suggests that one should assume that the EMV reader could collude with relay attackers in mounting fraudulent payments. Moreover, the current relay-counteracting EMV protocols [4,5] do not provide any evidence that the protocols were run correctly. So even if a complaint by a card-holder is made, it would not be possible to audit the EMV reader and see whether the distance-bounding checks had been performed. In this paper, we address these shortcomings. Concretely, our contributions are as follows:

I. We define the notion of *collusive relay attacks* to mean relay attacks in which the authentication-verifying party (EMV reader) can collude with a MiM relayer to mount a relay attack against an authentication and payment scheme; we define an attacker model and security definition for this new type of attack.

II. We present three new EMV protocols that defend against such a malicious reader. A summary of these protocols is given in Figure 1. A complicating factor is that bank cards have no accurate clock. Therefore, the card cannot distance-bound the reader. The complex EMV infrastructure also makes distance bounding between the bank and the card impractical. Our solutions show how adding a TPM as a hardware root of trust on-board the the reader can solve these issues.

III. We discuss our design choices, and provide a high-level argument w.r.t. the resistance to collusive relaying of the EMV protocols we propose.

2 Background & Foundational Aspects

Contactless EMV. Past work [4] has showed an effective relay attack against contactless EMV protocols, and suggested a version of contactless EMV called *PaySafe* that deters relay attacks. Following this, the main idea of the PaySafe protocol was added to the MasterCard specification (EMV contactless specifications v3.1 [5]) and yielded MasterCard's Relay Resistant Protocol (RRP). Part of this protocol is shown in Figure 2.

As with all EMV protocols, the card includes: (1) a private key Pv_C; (2) a symmetric key K_M that it shares with the bank; (3) a certificate chain $Cert_{Pv_{CA}}(Pub_C)$ for the card's public key Pub_C. The reader has the public key Pub_{CA} of the Certificate Authority, and so can extract and verify the card's public key. RRP starts with a setup phase (not shown in Figure 2), in which the reader asks the card what protocols it supports and selects one to run. The card and reader then generate single-use random numbers N_C and UN, respectively. The reader then sends an "EXCHANGE RELAY RESISTANCE DATA" command to the card, which contains the nonce UN. The card immediately replies with its own nonce N_C, and the reader times this round trip time. The card also provides timing information, which tells the reader how long this exchange should take. The reader compares the time taken with the timing information on the card. If

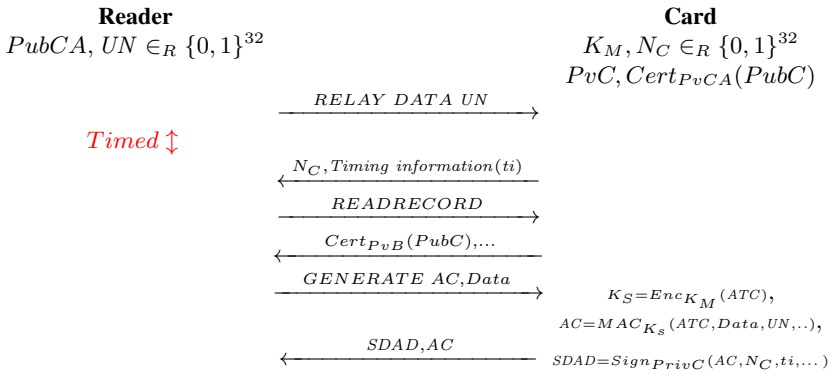

Fig. 2: MasterCard's Relay-Protected EMV

the time taken was too long, the reader stops the transaction as a suspected relay attack. Otherwise, the reader requests that the card generates a "cryptogram" (a.k.a. AC). The card uses the unique key K_M, which it shares with the bank, to encrypt its application transaction counter ATC (which equals the number of times the card has been used). This encryption equates to a session-key denoted K_S. The cryptogram AC is a MAC keyed with K_S of data including the ATC, the nonce UN, and the transaction information. As the reader cannot check the AC, the card generates the "Signed Dynamic Application Data ($SDAD$)": the card's signature on a message including UN, amount, currency, ATC, N_C. The reader checks the $SDAD$ before accepting the payment.

On TPMs. The Trusted Platform Module (TPM 2.0) is a hardware root of trust (see `https://trustedcomputinggroup.org`). It provides two measures of time: one is "`Clock`" (see page 205 of [8]) and the other is "`Time`" (see page 176 of [8]). `Time` is a 64-bit milliseconds count from when the TPM was powered up. `Clock` shows the real time; this is set when the TPM is created and must be "accurate even if there is no reliable external clock" [8]. The `TPM2_GetTime()` command takes the handle for a signature scheme and some input, and it returns a signature over `TPM_AttestedTime`=(`Clock`, `Time`) and the input. As such, `TPM2_GetTime()` can produce a signed version of a timestamped nonce, with attested time-information.

Attacks onto `TPM_AttestedTime` are mainly relevant w.r.t. the TPM `Clock` (see 36.3 and 36.6 [8]), as this has a non-volatile dimension, unlike `Time`. Notably, if the TPM is powered down, the `Clock` value is correct when the TPM reboots. The threats w.r.t. `Clock` documented by TCG, are as follows: (a) if adversaries can manipulate external software and local clocks like the CMOS clock on PC platforms, but if the TPM is not physically attacked, then the `Clock`'s accuracy (w.r.t. a small deviation from real time) is assumed to remain within "acceptable tolerance" (see page 206 of [8]); (b) the `Clock` value can only be deviated forward, i.e., it cannot be rewound.

3 System Setup, Threat Model & Security Requirements

Protocol Entities. Past distance-bounding work has involved a "*prover*" who demonstrates to a "*verifier*" that it is close (and possibly authenticates too). Our framework is different to this past work on DB. Rather than the two entities of the classic DB model, we have four entities in our setting: a "*card*" that interacts with an "*EMV reader*" in a

DB-fashion, the EMV reader will have an onboard hardware root of trust (a "*TPM*"), and the reader will send evidence for the transaction to the "*bank system backend*".

General Infrastructure & PKI. Our protocols use EMV's existing Public Key Infrastructure (PKI), augmented to support TPMs inside the readers. We assume that Certificate Authorities (CAs) have issued certificates on the TPMs' endorsement keys, that the banks, cards and EMV-related authorities have access to the right key-chains/certificate-chains to verify all certificates and, notably, first extract the TPMs endorsement keys. These endorsement keys are then use to verify other certificates sent by the TPM, e.g., certificating the public counterpart of a TPM's signing key. In this way, the bank and cards can, for instance, verify signatures issued by the TPM via a full chain-of-trust, up to the CAs.

Our Participants' & Communication Model. Between any card and any reader, we assume that all messages (irrespective of their bit-length) travel at an a-priori fixed constant speed, which is also the maximum speed of these radio interfaces. We assume that communication between the reader's software and the onboard TPM happens also at the maximum speed of the link between the two. We also assume that cards and readers can run several concurrent executions of the protocols. Such *honest* communication is possible if the card and the reader are no further than an a-priori fixed distance from one another.

Computation Model. Previous DB models have assumed a single, static RTT bound for all devices. However, our protocols (and MasterCard's RRP) use a card specific time bound. To help us formalise this we make the following two definitions:

Definition 1 (Proximity-checking Phase). *The* proximity-checking phase *of a protocol is an exchange of challenges and responses, which is timed by the challenger.*

Definition 2 (Card Time-Bounding Functions $\mathtt{t}_d$(cardID) and $\mathtt{t}$(cardID)). *We call $\mathtt{t}$(cardID) a* time-bounding *function; it maps a card identified by $cardID$ to the time, in time units, taken for that card to perform the computational part of the timed phase.*

*We call $\mathtt{t}_d$(cardID) a d-*time-bounding *function; it defines the duration of the proximity-checking phase when executed by a card identified by $cardID$ and physically found at a distance no larger than d from the reader. We write just t_d, when $cardID$ is implicit.*

Typically, $\mathtt{t}_d$(cardID) $=$ $\mathtt{t}$(cardID)$+$"time for all messages of the proximity-checking phase to travel distance $(2 \times d)$". We now define DB protocols with variable time limits.

Definition 3 (Contactless EMV Protocol with Proximity-checking Phase of Distance-Bound d). *A* contactless EMV protocol with proximity-checking phase of distance-bound d *(or, for short, contactless EMV protocol with distance-bound d) is a protocol between the EMV entities card, EMV reader and bank system backend, that has a proximity-checking phase. The protocol has additional parameters defined by the time-bounding function $t(cardID)$ and the d-time-bounding function $t_d(cardID)$, for each $cardID$. The reader side of the protocol may make use of a TPM. One of the EMV entities checks that the time recorded for the proximity-checking phase is inline with $t_d(cardID)$. If this is not the case, then the protocol finishes unsuccessfully.*

Definition 4 (Correct Execution). *Consider a contactless EMV protocol with proximity-checking phase of distance-bound d. If all entities in the system follow the protocol and*

the distance between the $card$ and the $EMVreader$ is no larger than d, then the protocol finishes successfully and a correct cryptogram AC for a payment will be issued by the card, and it will be eventually accepted by the bank.

Our Attacker Model. Combining DB [3, 6] and EMV models [2], we assume an attacker that also completely controls a number of cards, including all their key material. The attacker can act and use the corrupted card's keys at any location. Unlike in previous DB models [6], the attacker can know the readers' key/secret material and can control the software on the readers to make it perform arbitrary actions.

We assume that the TPM is as secure as is claimed in its specification, see page 3, Section 2. That is, 1. our attacker cannot tamper with the initial setup of the TPM's `Clock`; 2. our attacker cannot mount any physical attack on the TPM's time-reporting `TPM-AttestedTime`=(`Clock`, `Time`); 3. our attacker can deviate the TPM's `Clock` only by making it go forward w.r.t. to the real time by a negligible fraction.

We assume the attacker cannot make the messages travel faster between the card and the reader, nor on the link between the reader and the TPM; recall from the communication model that both these interfaces are respectively set at their the maximum communication speeds (which is also constant).

We assume all cryptographic primitives used in the EMV protocol are secure w.r.t. their respective threat-models, e.g., signatures are *unforgeable* etc.

Our Security Requirements. The main aim of our attacker is to trick the bank system backend to accept an AC generated by a card that was not in close proximity with a reader. We formalise this as:

Definition 5 (Resistance to Collusive Relaying). *A contactless EMV protocol of distance-bound d is* resistant to collusive-relaying *if, for any attacker in the threat and communication model above, for any payment AC that the bank system backend accepts from a card that is not controlled by the attacker, the card must have been within distance d of the reader for the time bounding phase that lead to the generation of the AC.*

4 EMV Protocols Resistant to Collusive-Relaying

4.1 `PayCCR`: A Protocol Compatible with the Current Banking Backend

Our first protocol, *`PayCCR`*, is shown in Figure 3. It modifies the EMV protocol on the card and the EMV reader's side, yet the bank system backend remains unchanged from the current standard. As with MasterCard's RRP protocol, the time bound $\mathtt{t}_d$(cardID) to be enforced for the proximity-checking phase is embedded in each card. Below, we write this bound as t_d. This time bound is chosen when the card is created, based on its processing speed, to ensure that the card and EMV reader are less than d distance from each other. The full protocol starts off with a standard EMV set up phase, in which the payment app is selected. The reader starts the proximity-checking stage of the protocol by sending the card a certificate chain for the TPM's public part of the signing key.

The EMV reader will then send a nonce N_R to the TPM to be timestamped. The TPM receives this bitstring N_R passed to the $TPM2\_GetTime$ command, the TPM timestamps it with `TPM-AttestedTime`, and using the randomised signing algorithm ECDSA produces the signature σ_1. Then, the EMV reader forwards σ_1 to the

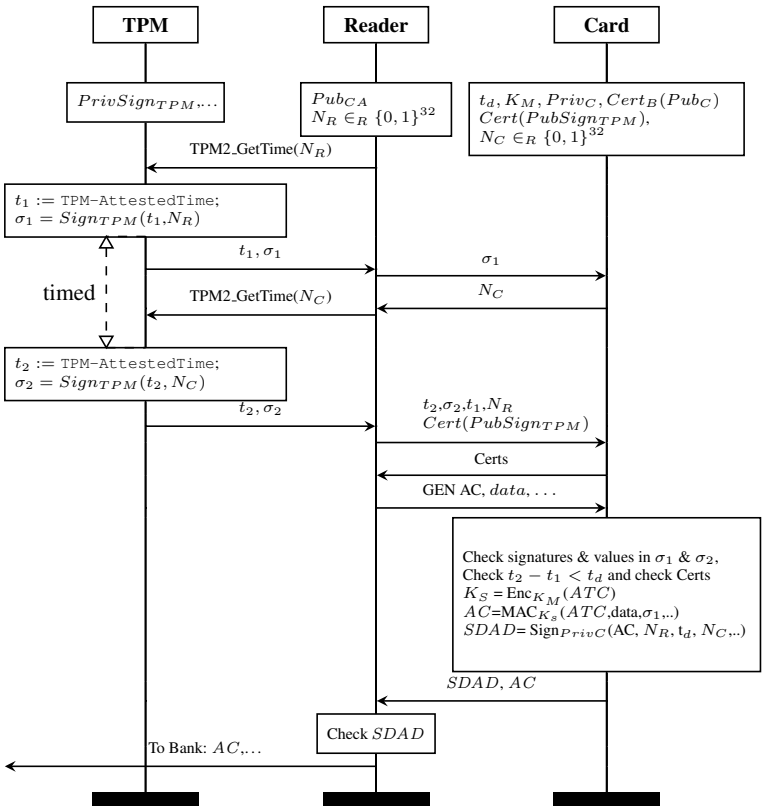

Fig. 3: PayCCR: Protection against Collusive-Relay and No Changes to the Bank's Backend

card. This should be done by the reader at the maximum speed of the interface, i.e., as each bit is received from the TPM it should be forwarded to the NFC interface. We allow the nonces to be split into bytes and the time stamping and nonce exchange to be repeated four times, once per byte. The average of the four time differences would be compared with t_d.

The nonce N_C is pre-generated, thus making the reply time fast. The TPM timestamps N_C (producing σ_2), and the reader sends the signature σ_2 to the card. The card sends its certificates to the EMV reader, which then asks the card to generate the AC to complete a payment. Before generating the AC the card checks the TPM certificate provided by the EMV reader, verifies the signatures on the timestamps σ_1 & σ_2, and ascertains that the time bound is less than its allowed maximum value t_d. If these checks pass, then the card generates an AC and $SDAD$, which are sent to bank via the reader, and checked by the bank as normal. If any of the card's checks fail, then the card sends a declined message to the reader and aborts.

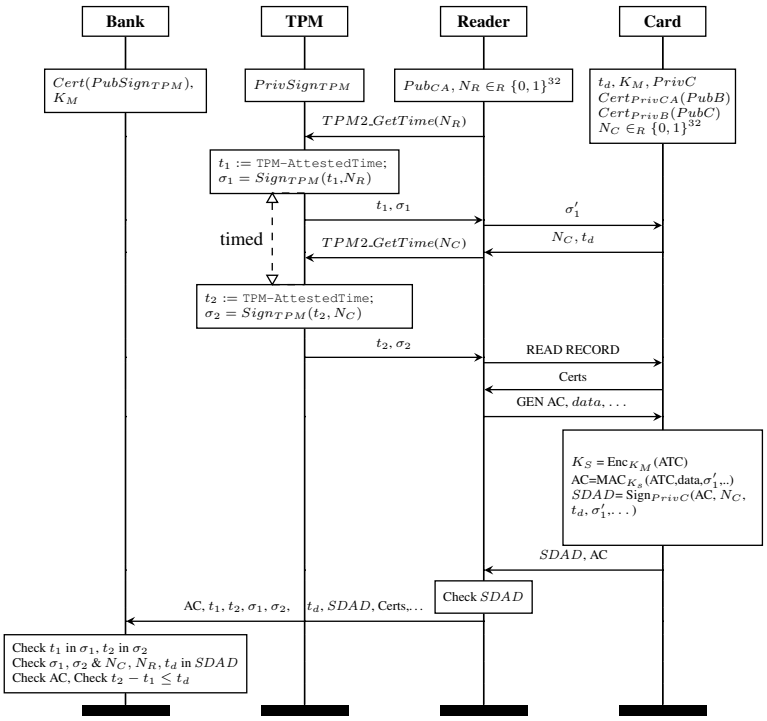

Fig. 4: PayBCR: Contactless EMV Protection with No Changes to the Card

Discussion. This protocol, and the others herein, make two main assumptions:
1. that a time bound t_d for each card can be apriori set;
2. dishonest reader cannot receive and/or send proximity-checking-phase messages faster than an honest reader, except for an insignificant amount.

We detail on the second assumption above. The assumption, present in our threat model in the previous section, implies that the TPM and NFC APIs used must run at their maximum speeds. Even so, a dishonest reader may gain some advantage by running at a faster clock speed than an honest reader. However, as the only effective timed action undertaken by the reader is forwarding bits from the TPM interface to the NFC interface, an overclocked reader can only gain an advantage in the order of nanoseconds (referred to in assumption 2 as an "insignificant amount"). Such an advantage would translate into a theoretical relay attack (including w.r.t. our definition), however –in practice– it would be a relay over a distance larger than the bound only by a few centimetres. We could tighten Def. 5 to exclude such practically irrelevant attacks. Instead, we choose to just discard it out right, on grounds of it being insignificant. In the conclusions, we discuss other ways of deterring/detecting readers running in an overclocked mode in the proximity-checking phase.

4.2 PayBCR: A Collusive-Relay Resistant Protocol Compatible with RRP Cards

Our second protocol, *PayBCR*, does not modify the card's side w.r.t. the current RRP protocol [5]. PayBCR achieves this in three steps: (1) it uses a timestamped signature from the TPM instead of what is now the reader's nonce in the EMV protocol v3.1; (2) the TPM timestamps the card nonce; (3) both timestamps are passed to the bank along with the AC, and the bank can check the difference between the timestamps to ensure the card and EMV reader where close. Additionally, the timing information ti on the current RRP cards is used as our time bound t_d. As with RRP, storing the time bound on the card and signing it avoids the bank having to maintain a look up table of all card's time bounds. The full protocol is shown in Figure 4. The EMV reader sends its nonce N_R to the TPM to be time stamped. The signature σ_1 from the TPM is sent to the card instead of the first nonce UN in the current RRP. To keep the protocol compliant with the current contactless EMV protocol in Figure 2, this bitstring we send to the card should be shorter, this is achieved by truncating σ_1; this truncation we denote as σ_1'.

Like in the current relay-protecting EMV protocol, the card replies with its nonce N_C. The card's nonce is immediately sent to the TPM to be timestamped. The protocol continues in nearly the same way as the current relay-protecting EMV protocol. The $SDAD$ now signs the AC, the timing information and σ_1' (in place of UN), this along with the card's time-bound t_d, σ_1, σ_2, t_1 and t_2 and the AC are sent to the bank. The bank will check that the TPM's signed timestamps match the nonce values used in the AC, the timing information is correctly signed, and the time difference between the nonces is less than this time bound. Other details, not in Figure 4, are either as in protocol PayCCR or are self-explanatory.

PayBCRv2: As a variant of our PayBCR, we can have the EMV readers store the TPM's signed timestamps σ_1 & σ_2, and the time values t_1 & t_2, the $SDAD$ and card certificates, and not send them to the bank system backend, i.e., the protocol would be backwards compatible with both the current standards for the bank system backend and cards; only changes to the EMV readers are required.

Such a variant would entail that a collusive-relay attack could not be stopped in real time. Rather this version of the protocol would be suited for when a card owner raises a complaint, or for the bank to detect possible fraud a-posteriori. At such a point, the EMV reader would be audited and all of the transactions would be checked.

Discussion. This protocol variant would be much easier to deploy than those discussed above, EMV reader manufacturers could add this protection unilaterally, without needing to make any changes to the current EMV specifications or the bank system backend. Making changes to the specifications for cards would be a slow process requiring input from many stakeholders, and making changes to the bank system backend would be expensive, due to the dedicated hardware banks generally use. Therefore, this protocol variant has a clear advantage over the others. The disadvantage of this protocol variant is that collusive-relay attacks could still be carried out and only detected during an audit. However, this is in keeping with much of the rest of the EMV security model that allows some fraud and aims to detect, roll back or refund it after the event. Our protocol would make any malicious interference by the EMV reader in the proximity-checking phase detectable, meaning that the bank could refuse payments to the reader,

if the audit information was missing or did not check out, so removing all motivation for this attack. Therefore, while the protections provided by the PayBCRv2 protocol are the weakest, it is perhaps the most practical to introduce.

Lastly, whilst our protocols implicitly timestamp payments, in this work we do *not* investigate links between correct/secure payments and their associated transaction time.

5 High-level Security Assessment

Our protocols do not alter any security property of the current contactless EMV; the authentication properties of our protocols follow from the basic EMV protocol. As in EMV, the freshness from the card and the reader stop replay attacks. The AC is generated based on a key shared only between the card and the bank, so the bank can be sure that this came from a card; the reader gets similar guarantees from the signed $SDAD$.

Assume protocol PayCCR is run in the presence of an arbitrary attacker in our model, and an AC is sent out by a card not controlled by the attacker. In PayCCR if the bank system backend accepted an AC then:

1. The backend checks the AC based on the card key. So, the AC must have come from that card (which is not controlled by the attacker), therefore this card will have executed its algorithm, i.e., performed the required checks.
2. The card checks the certificate for the TPM's signing key. Therefore, the card can be sure of the timestamps signed by the TPM.
3. Since the card checks that σ_1 includes the timestamp t_1, the card can be sure that the σ_1 message originated at the EMV reader's TPM at time t_1.
4. The card will only broadcast N_C after it has received σ_1.
5. The card checks that σ_2 includes its nonce N_C and the time t_2, therefore it can be sure that the reader received the nonce N_C before time t_2.
6. Together (3), (4) and (5) ensure that the RTT of the messages σ_1 and N_C was at least $t_2 - t_1$ and that these messages went between the card and the TPM.
7. The card knows its time-bound t_d. So, checking that $t_2 - t_1 < t_d$ ensures that the card was within distance d of the reader, which gives us resistance to collusive-relaying (Def. 5).

We now place ourselves in the setting where the protocol PayBCR is run in the presence of an arbitrary attacker in our model and an AC is sent out by a card not controlled by the attacker. For our second protocol, PayBCR, recall that the checks are carried out by the bank system backend, therefore the following reasoning applies:

1. Checking that σ_1' and N_C are in the $SDAD$ ensures that the reader/attacker sent σ_1' to the card, and that the card thereafter used the nonce N_C.
2. As σ_1 is a high-entropy, randomised signature, checking that σ_1 signs t_1 means that σ_1 was generated at time t_1 and cannot have been sent to the card before this.
3. By looking at the timestamps in σ_1 and in σ_2, and at the fact that σ_2 also signs N_C, the bank system backend will know that N_C was only broadcast by the card after t_1 (and in fact after σ_1 was received) and before t_2. The bank system backend also checks this against $SDAD$, which signs N_C, σ_1.
4. Together (2), (3) and (4) guarantee that the round trip time between the card and the EMV reader's TPM was less than $t_2 - t_1$.

5. Checking that t_d is in the $SDAD$ ensures that the correct time-bound for the specific card is used in the checks.
6. Checking that $t_2 - t_1 < t_d$ together with (5) and (6) ensures that the distance between the card and the reader was within distance d of each other.

6 Conclusions

In this paper, we presented three protocols that show how –by using a TPM– rogue readers can be stopped from subverting the relay-detecting checks in contactless EMV. The three protocols with different levels of compatibility with the current EMV framework. We also put forward an attacker model (in line with using TPMs as roots of trust, considering dishonest EMV readers, etc.) and a new security definition that protects against reader-assisted relaying in EMV protocols.

In one line of future work, we wish to develop a new, fully-fledged symbolic formalism and a provably-secure models that can be used to prove the correctness and security w.r.t. collusive relaying.

Moreover, we plan to implement our protocols to show that our proposed use of the TPM can lead to a workable EMV protocol with such protections against strong relaying. For `PayCCR`, we will measure the time it takes for ubiquitous smart-cards to verify different randomised signatures.

We will investigate the second assumption of our designs, i.e., that reader's computations in the proximity-checking phase being kept at a constant amount. A step further to investigate is to certify the read/write speed of the readers via a TPM (i.e., using `TPM_GetQuote` or other host-attestation methods); this type of method can add security guarantees but it would clearly require further checks by the bank.

Acknowledgments. The authors acknowledge the support of the NCSC-funded "TimeTrust" project. The authors also thank all anonymous reviewers, as well as Urs Hengartner for helpful comments. Also, Ioana Boureanu thanks Anda Anda for interesting discussions on this topic.

References

1. S. Brands and D. Chaum. Distance-bounding protocols. In *EUROCRYPT '93*, 1994.
2. C. Brzuska, N. P. Smart, B. Warinschi, and G. J. Watson. An analysis of the EMV channel establishment protocol. In *Conference on Computer & communications security*, 2013.
3. T. Chothia, J. de Ruiter, and B. Smyth. Modelling and analysis of a hierarchy of distance bounding attacks. In *USENIX Security'18: 27th USENIX Security Symposium*, 2018.
4. 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)*, LNCS, 2015.
5. EMVCo. Book C-2 kernel 2 specification v2.7. EMV contactless specifications for payment system, Feb, 2018.
6. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical and Provably Secure Distance-Bounding. *Journal of Computer Security*, 23(2):229–257, 2015.
7. X. Shu, K. Tian, A. Ciambrone, and D. Yao. Breaking the target: An analysis of target data breach and lessons learned. *CoRR*, abs/1701.04940, 2017.
8. Trusted Computing Group. Trusted Platform Module Library Family 2.0, Specification - Part 1: Architecture, Revision 1.38 and Part 3: Commands, Revision 1.38, 2016.