Provable-Security Model for Strong Proximity-based Attacks – With Application to Contactless Payments –

Ioana Boureanu
Liqun Chen
Sam Ivey
i.boureanu@surrey.ac.uk,liqun.chen@surrey.ac.uk,s.ivey@surrey.ac.uk
University of Surrey, Surrey Centre for Cyber Security (SCCS)
Guidford, UK

ABSTRACT

In Mastercard's contactless payment protocol called RRP (Relay Resistant Protocol), the reader is measuring the round-trip times of the message-exchanges between itself and the card, to see if they do not take too long. If they do take longer than expected, a relay attack would be suspected and the transaction should be dropped. A recent paper of Financial Crypto 2019 (FC19) raises some questions w.r.t. this type of relay-protection in contactless payments. Namely, the authors point out that the reader has no incentive to protect against relaying, as it stands to gain from illicit payments. The paper defines the notion of such a rogue reader colluding with a MiM attacker, specifically in the context of contactless payments; the paper dubs this as *collusive relaying*. Two new protocols, PayBCR and PayCCR, which are closely based on Mastercard's RRP and aim to achieve resistance against collusive relaying, are presented therein. Yet, in the FC19 paper, there is no formal treatment of the collusive-relaying notion or of the security of the protocols.

In this paper, we first lift the FC19 notions out of the specifics of RRP-based payments – to the generic case of distance bounding. Thus, we set to answer the wider question of what it would mean to catch if RTT-measuring parties (readers, cards, or others) cheat and collude with proximity-based attackers (i.e., relayers or other types). To this end, we give a new distance-bounding primitive (*validated distance-bounding*) and two new security notions: *strong relaying* and *strong distance-fraud*. We also provide a formal model that, for the first time in distance-bounding, caters for dishonest RTT-measurers. In this model, we prove that the new contactless payments in the FC19 paper, PayBCR and PayCCR attain security w.r.t. strong relaying. Finally, we define one other primitive (*validated and audited distance-bounding*), which, in fact, emulates more closely the PayCCR protocol in the Financial Crypto 2019 paper; this is because, contrary to the line introducing them, we note that PayBCR and PayCCR in fact differ in construction and security guarantees especially in those that go past relaying and into authentication.

ASIA CCS '20, October 5–9, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-6750-9/20/10…$15.00
https://doi.org/10.1145/3320269.3384748

ACM Reference Format:
Ioana Boureanu, Liqun Chen, and Sam Ivey. 2020. Provable-Security Model for Strong Proximity-based Attacks – With Application to Contactless Payments –. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS '20), October 5–9, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3320269.3384748

1 INTRODUCTION

In relay attacks, a malicious party forwards communications between two legitimate parties, without them knowing and with the aim to have some gain. In 2015, EMV (Europay Mastercard and Visa), which is the most widely used electronic payment protocols was shown [8] to be susceptible to these relay attacks. In turn, this implies that contactless cards are abused by attackers to unwillingly make payments to a far-away EMV reader. As such, in 2016, Mastercard enhanced its contactless EMV protocol, called PayPass, with a relay-protection mechanism. The resulting protocol was dubbed RRP(relay-resistant protocol). For relay-protection, in RRP, the terminal enforces an upper bound on the round trip times (RTTs) of the messages it exchanges with the card. This is a RTT-based method of relay-deterrence, widely known as *proximity checking* or *distance bounding* (DB) [6].

Recent attention has been given to RRP. Namely, in [7], the authors note that the RTT-measurements sitting with the EMV reader may not fit with the incentives of the latter; specifically, some EMV terminals would be ready to take illicit, relayed payments, as they only stand to gain from it anyway. Moreover, the issuing-bank in RRP does not get any proof of the RTT-checks, so a rogue terminal as per the above can "walk free". Thus, [7], did propose new protocols based on RRP, with the view that these protocols would catch such a cheating reader that would potentially collude with a relay attacker.

The authors of [7] focused purely on RRP (as opposed to other DB/payment protocols) and on one definition capturing the potential collusion between the rogue reader and the relay-attacker. This definition was called *collusive relaying*, and it is given informally. Moreover, this definition is (again) cast in the strict setting of EMV. Concretely, the security against collusive relaying says that if the issuing-bank is authenticating a contactless card during a payment, then that card must have been close to the reader (implicitly the reader who asked for the payment be taken). Two things strike w.r.t. this definition. Firstly, it mixes a property of authentication and one of proximity in ways that are not totally clear. For instance, it is not clear which inputs the bank ought to receive w.r.t. RTT-checks, if any. This generality could be fine for a generic primitive, but it

is arguably too loose given that it is applied to a specific case, like EMV. Secondly and on a more pedantic note, the name *collusive relaying* via the lens of the given definition is misleading, as relaying per se is an attack against proximity-checking not against authentication [11]. For an attack against authentication, we generally speak of MiM (men in the middle) that do more than just relaying. So, it appears that [7] aimed to introduced not just security against relay-facilitating collusions, but also/rather security against MiM-facilitating collusion.

Also, the general question of an actual primitive that would augment distance-bounding to protect against cheating by the RTT-measuring parties was not asked by [7]. In this vain, [7] stops short of asking itself if there are other collusive attacks in DB which would align to collusive relaying. I.e., could an attacker collude with an RTT-measuring card to mount a sort of *collusive distance-fraud*[1].

Last but not least, [7] does not include a formal model, not even one just for collusive relaying over EMV.

Research Questions. This paper aims to bridge these gaps. It aims to "peel" back to layer to the original and generic idea in [7], and answer these questions:
Q1: "If the RTT-measuring party in DB is corrupted, can we still get some security against a strong form of relaying or of other proximity-based attacks, whereby –in the attack– the corrupted entity could side with the main attacker?"
Q2: "Can we formally prove this type of security for the protocols in [7]?
Q3: "What primitive would encapsulate this security property best?"
Q4: "What would the security model for this primitive be?"
Q5: "How do the security definitions yielded here compare with the collusive-relaying notion introduced in [7]?".

Note. Consider the presence of rogue RTT-measurers who may collude with other attackers. In this context, we would like to point out that it is of practical interest to be first looking at collusions just w.r.t. relaying (as opposed to collusions over relaying and authentication as per the albeit informal endeavours in [7]). Indeed, relaying is a simple attack to put in place (which subverts no cryptography), as opposed to authentication forgeries. That is why we pursue the relaying line primarily, and only in Section 5 move towards including collusive attacks over authentication as well.

Contributions & Structure. In this paper, we answer the above questions as follows.

(1) Answering Q3 above, we introduce a generic, augmented DB primitive, called ***validated distance-bounding (v-DB)***. In this primitive, a party is mandated to recheck the RTT-measurements, with the view to catch corruption therein. See Section 3.
(2) Answering Q4 above, we define a security model that allows for the RTT-measuring party to be malicious. Incidentally, this also means that it allows for a reader to be malicious (if the reader does the RTT). To the best of our knowledge, this is the first formal model of this type in the DB literature. See Section 4.
(3) Answering Q1 above, v-DB protocols, we define the security properties of ***strong relaying*** and ***strong distance-fraud***. See Subsection 4.4.
(4) Answering Q2 above, we prove that the protocols in [7] attain ***strong relaying***. ***Strong distance-fraud*** does not apply to them. See Subsection 4.5, and Appendix A for the actual proofs.
(5) Answering Q5 above, we also introduce the primitive (***validated and audited distance-bounding (v-ADB) protocols***), and the property (***strong MiM v-ADB-security***) that would be closer to the notion of collusive relaying in [7]. See Section 5.

[1]Distance-fraud is a DB attack whereby a far-away card manages to make it look like it is close to the reader.

2 BACKGROUND & FOUNDATIONAL ASPECTS

2.1 Contactless Payment Protocols Designed around Collusive Relaying

In [7], the authors introduce two protocols PayCCR and PayBCR, both based on Mastercard's RRP. These protocols assume the addition of Trusted Platform Modules (TPMs) to the EMV infrastructure (i.e., to the PKI, in such a way that banks can check TPM certificates during EMV transactions). In this section, we recall relevant details of these protocols.

PayBCR & PayCCR: High-level Description. Both protocols in [7] enhance Mastercard's contactless-payment protocol with relay protection, i.e., RRP [12] . They do so by adding a TPM onboard the RRP reader. This TPM is called twice, each time to timestamp an input, such that the difference of the two timestamps closely approximates the roundtrip time (RTT) between the card and the reader. Moreover, PayCCR and PayBCR record this timestamping information, later to be used by the card or the issuing bank to re-verify the RTT measurements, alongside other checks each normally makes in RRP. On one hand, PayBCR does not modify the card side of RRP and thus it is the issuing bank who does the verification of the TPM's timestamps. On the other, PayCCR leaves the RRP reader-to-bank specifications unchanged, and it modifies the RRP card so that it re-checks the RTT measurements mediated by the TPM onboard the RRP reader.

Herein, we will mainly recall PayBCR. The reader is referred to [7] for details. PayBCR , shown in Figure 1, is tightly based on RRP. Firstly, the EMV reader sends its nonce N_R to the TPM to be timestamped. The TPM uses the *TPM2_GetTime* command to timestamp this nonce and it produces a randomised signature σ_1 on the timestamped nonce. The signature σ_1 from the TPM is sent to the card, in lieu of the nonce so-called *UN*in RRP. To keep PayBCR compliant with RRP, a truncation of σ_1 is sent to the card; this truncation is denoted as σ_1'. The card's response (N_C as per RRP) is sent to the TPM, which similarly yields a randomised signature σ_2. The *SDAD* signs the *AC*, the timing information and σ_1' (in place of *UN*). Finally, the card's RRP time-bound t_d, σ_1, σ_2, t_1 and t_2 and the *AC* are sent to the bank. With these, the bank can check the difference between the timestamps to ensure the card and EMV reader were close.

For completeness, we give the description of PayCCR too; see Figure 2. Its details are very similar to those of PayBCR, only that it

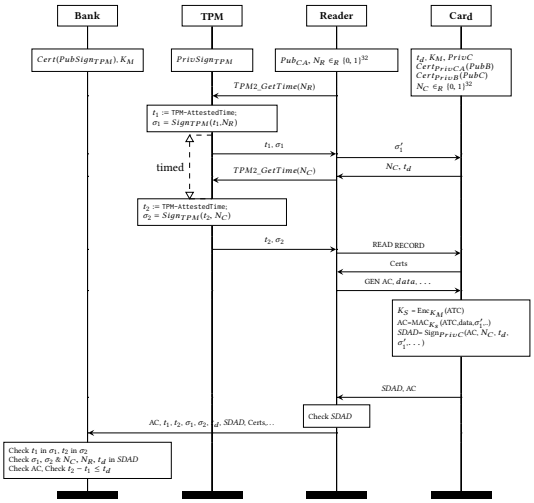

Figure 1: PayBCR [7]: Mastercard's RRP with Collusive-Relay Protection & No Changes to the Card

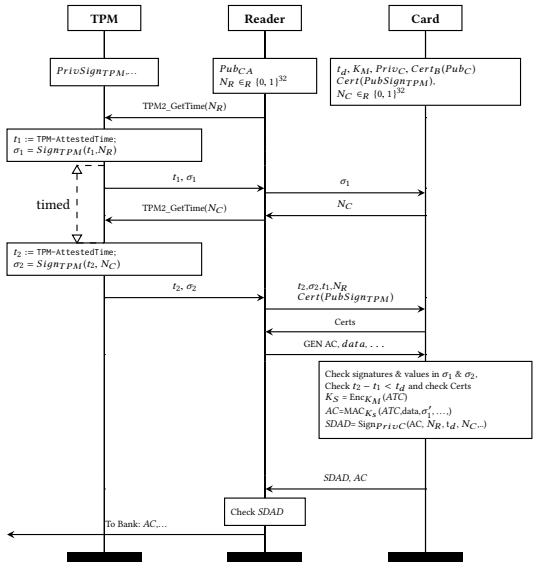

Figure 2: PayCCR [7]: Mastercard's RRP with Collusive-Relay Protection & No Changes to the Issuing Bank

is the card who does the verification of the timestamping signatures.

3 VALIDATED DISTANCE-BOUNDING PROTOCOLS

In this section, we introduce the notion of *validated distance-bounding (*v-DB*)*. We then discuss this definition w.r.t. existing lines in the field.

3.1 v-DB Definitions

This is an augmented DB protocol in the sense that the following two aspects are added to a "standard" DB protocol:

a. the proximity-checking algorithm is portable and, unlike in traditional DB, it can be added not only to the reader, but –alternatively– it can be added to the card;
b. the proximity-checking is validated in the sense that, in an v-DB protocol, after the proximity-checking phase is finished, a protocol party will re-verify that the proximity-checking was performed as expected.

In other words, the above means that –unlike in standard DB– in a v-DB protocol, the *reader* may or may not be the one to undertake the RTT measurements, but –no matter which protocol party performed the measurements– these are certainly re-checked by another algorithm in the protocol. We now formalise this below.

Definition 3.1. **Validated Distance-Bounding Protocols**. A *validated distance-bounding (*v-DB*) protocol* is a tuple $\Pi = (\mathcal{C}, \mathcal{R}, \mathcal{T}, \mathcal{PC}, \mathcal{W}, \mathbb{B})$, where $\mathbb{B}$ denotes the distance bound and $\mathcal{C}, \mathcal{R}, \mathcal{T}, \mathcal{PC}, \mathcal{W}$ are ppt[2]. algorithms as follows:

- $\mathcal{C}$ is the card algorithm and $\mathcal{R}$ is the reader algorithm in an unilateral authentication protocol where $\mathcal{C}$ authenticates to $\mathcal{R}$;
- $\mathcal{T}$ is a tamper-resistant, trusted execution environment;
- the tuple $(\mathcal{PC}, \mathcal{T})$ form a proximity-checking functionality: $(\mathcal{PC}, \mathcal{T})$ checks that $d(\mathcal{C}, \mathcal{R}) \leq \mathbb{B}$;
- the tuple $(\mathcal{PC}, \mathcal{T})$ is directly used by one authentication party: be it by $\mathcal{C}$ or by $\mathcal{R}$;
- $\mathcal{R}$ and $(\mathcal{PC}, \mathcal{T})$ respectively have public outputs Out_R and $Out_{(\mathcal{PC},\mathcal{T})}$ in $\{0, 1\}$ (success/failure of the authentication and proximity-checking respectively), as well as private outputs denoting their transcripts, denoted $\tau_{\mathcal{R}}$ and, resp., $\tau_{\mathcal{PC}}$;
- $\mathcal{W}$ is the *proximity validating* algorithm: given the private output of $(\mathcal{PC}, \mathcal{T})$, the algorithm $\mathcal{W}$ checks the correctness of public output of $(\mathcal{PC}, \mathcal{T})$.
- $\mathcal{W}$ has a public output Out_W.

Def. 3.1 says that a validated distance-bounding (v-DB) protocol is composed of two primitives. First, it comprises the authentication primitive that implements the mechanisms of C authenticating to R. Second, it includes the proximity-checking primitive facilitated by $(\mathcal{PC}, \mathcal{T})$, which is aimed to verify that the devices/parties running C and resp. R are at a distance of no more than $\mathbb{B}$ from one another. Note that the validating algorithm $\mathcal{W}$ solely re-verifies the proximity-checking, without the authentication side (i.e., $\mathcal{W}$ operates on the private output $(\mathcal{PC}, \mathcal{T})$ only, without that of $\mathcal{R}$).

Remark 1: W.r.t. above, we note:

[2]All measures of complexity and probability are asymptotic in a security parameter s.

(1) Since the authentication and proximity checking algorithms are separated, one can clearly differentiate the corruption of one or the other, thus yielding a finer threat model than in "standard DB" [1], and propose specialised security properties.
(2) Unlike in "standard DB", in v-DB, the proximity-checking can sit either with $\mathcal{C}$ or $\mathcal{R}$. Here, $\mathcal{W}$ only re-verifies the proximity-checking dimension of the protocol between $\mathcal{C}$ and $\mathcal{R}$. That means, that whilst we introduced v-DB on top of unilateral authentication, in fact v-DB is agnostic of the underlying protocol run by $\mathcal{C}$ and $\mathcal{R}$ and that v-DB would work the same if this latter protocol were something other than authentication.

We now move to the notion of a validated distance-bounding system, which intuitively takes a v-DB protocol to the implementation level. To this end, some setup phases are run, algorithms are instantiated and loaded onto devices, and –importantly– the proximity-checking algorithm is fixed on one side: on the card-implementing side, or on the reader-implementing side. We formalise this in Def. 3.2 below.

Definition 3.2. **Validated Distance-Bounding (v-DB) Systems**. Given a v-DB protocol Π and ppt. algorithm X, a *validated distance-bounding system* Π^{real} (v-DB *system*, for short) is a concrete representation of v-DB and X, written v-DB^{real} $=(C, PC, T, R, W, X, \mathbb{B})$, as follows:
– the bound $\mathbb{B}$ is instantiated,
– the card and reader algorithms are set up[3] correctly to run the unilateral authentication protocol in v-DB,
– the proximity-checking algorithm $(\mathcal{PC}, \mathcal{T})$ is set up[4] correctly and is coupled with the card algorithm or with the reader algorithm,
– the validating algorithm $\mathcal{W}$ is set up[5] correctly and coupled with the card algorithm or with the reader algorithm opposite to how the proximity-checking algorithm is coupled, or –alternatively– it is coupled with X,
– after this coupling and setup, all algorithms in Π are loaded onto communicating devices.

By enumerating where $(\mathcal{PC}, \mathcal{T})$ and $\mathcal{W}$ can "sit" in a v-DB system, we obtain the following classes of v-DB systems.

Definition 3.3. **Classes of v-DB Systems**. Consider a v-DB system denoted as v-DB^{real} $=(C, PC, T, R, W, X, \mathbb{B})$ be . If the proximity-checking algorithm is coupled with the card and the validating algorithm is coupled with the reader, we write Π^{real} $=([C, PC, T], [R, W], \mathbb{B})$, and we call this *card-checked* and *reader-validated* distance-bounding system. The $[C, PC, T]$ tuple is called a *card coupling*.

If the proximity-checking algorithm is coupled with the reader and the validating algorithm is coupled with the card, we write Π^{real} $=([R, PC, T], [C, W], \mathbb{B})$, and we call this *reader-checked* and *card-validated* distance-bounding system. The $[R, PC, T]$ tuple is called a *reader coupling*.

If the proximity-checking algorithm is coupled with the card and the validating algorithm is coupled with X, we write Π^{real} $=([C, PC, T], R, [W, X], \mathbb{B})$, and we call this *card-checked* and *X-validated* distance-bounding system.

If the proximity-checking algorithm is coupled with the reader and the validating algorithm is coupled with X, we write Π^{real} $=([R, PC, T], C, [W, X], \mathbb{B})$, and we call this *reader-checked* and *X-validated* distance-bounding system.

The Scope of* v-DB *Systems. With Def. 3.3 in place, one can now see that v-DB systems are paramount in cases where access to services are provided based on proximity checking, and where the "traditional" party ascertaining proximity may have incentives to misbehave and not execute its role correctly. In such cases, the output by T will aid another party in the protocol to run W as to re-verify that the proximity-checking was indeed performed, even by a potentially corrupt proximity-checker. We detail this below.

In reader-checked systems, Def. 3.2 aims to formalise the following: a v-DB protocol is run between a card and a reader, and the reader does the proximity-checking PC assisted by a trusted execution environment denoted T. In this case, the card or a 3rd party X re-validates that the proximity-checking was done correctly. Intuitively, in this case, the trusted execution environment T will prevent the reader from cheating and from not performing the proximity-checking PC correctly. An example of where a reader may wish to cheat and not perform the proximity-checking is that where its core function is orthogonal to the closeness of the card: i.e., in EMV, the reader just wants to take a payment, and would arguably still take payment from a card that was afar.

In card-checked systems, Def. 3.2 formalises a situation akin to that of reader-checked systems, only that –in this case– the roles of the card and reader are inverted. That is to say, imagine that a card is tasked with checking its proximity to the reader that is mobile. Assume this card has an incentive to lie about the proximity-checking (e.g., in order to allow a far-away reader to authenticate it). Then, in card-checked v-DB systems, we add T on board the card and the re-verifier W (which can sit with the reader or a 3rd party) will use T's outputs to detect that such a corrupted card may wish to fault the system.

3.2 v-DB's Place in the Field of Distance-Bounding

Examples of* v-DB *Systems & Related Definitions. The formalisation of v-DB systems is introduced for the first time in this paper. However, we can point to two existent systems that fit this description. PayBCR in [7] is a *reader-checked* and *bank-validated* v-DB system, whilst PayCCR in the same [7] is a *reader-checked* and *card-validated* v-DB system.

We are not aware of card-checked v-DB. Whilst there exist protocols where the card does check the proximity to the reader (e.g., mutual distance-bounding [2]), in these cases there is no validation (via another algorithm W) of this checking by the card.

Definitions Relating to* v-DB *Systems. Vaudenay and Kilinc, in [15], augment distance-bounding protocols by adding a Hardware Security Module (HSM) to cards, which is akin to Def. 3.2 coupling T and C. That said, the Kilinc-Vaudenay systems yielded are not card-checked (i.e., PC is not on the card side) and, moreover, no validating algorithm is present. In other words, the security

[3] All PKI involved or any cryptographic keys pre-shared are all set up.
[4] E.g., $\mathcal{PC}$ is set up with the same $\mathbb{B}$ as instantiated.
[5] E.g., $\mathcal{W}$ is set up with the same $\mathbb{B}$ as instantiated.

reasons and mechanisms of [15] are different: therein, cards are augmented with HSMs mainly to protect against a threat called terrorist-fraud [9] and not with our purpose of aiding to the proximity-checking mechanism per se, and –unlike herein – there is no dimension of re-validation of said proximity-checking by another party.

On a similar note, in [10], the authors look at mixing proximity-checking with roots of trust, akin to Def. 3.2 putting together T and PC. There are two notable differences between [10] and our definitions. (1) In [10], software roots of trust and –in fact– specific ones (i.e., Intel SGX) are envisaged to stand in for T and the proximity-checking algorithm is also fixed; this is clearly not the case in this work. (2) In [10], there is no aspect of re-validation of said proximity-checking by another party.

We sum up that Defs. 3.1 and 3.2 introduce a new primitive/protocol that was not defined before; however, we can see that they do offer *some* formalisation[6] for the contactless payment protocols published at Financial Crypto 2019, in [7].

4 A FORMAL MODEL FOR VALIDATED DISTANCE BOUNDING

In this section, we will present a formalism that captures the execution and threat models for v-DB protocols and systems.

In this model, we will be able to *formally* express for the first time certain fine-grained corruptions, which will be the basis of putting forward new security properties. For instance, in a v-DB system $\Pi^{\text{real}} = ([R, PC, T], [C, W], X, \mathbb{B})$, a man-in-the-middle (MiM) attacker can corrupt the R and/or the PC sides of the reader-coupling $[R, PC, T]$ and, as such, this yields collusive attacks whereby the corrupted reader can strengthen MiM attacks.

In the model that we are about to introduce, it is possible to capture other types of strong, collusive attacks that had not been introduced in the past. One example of that is our Definition 4.4 of strong distance-fraud, as well as the new attacks such as the ones we discuss in Section 5.

4.1 Execution Model.

v-DB *Parties.* In this paragraph, we detail part of the setup in a v-DB system Π^{real}, presented in Def. 3.2. Let $\Pi = (C, \mathcal{R}, \mathcal{T}, \mathcal{PC}, \mathcal{W}, \mathbb{B})$ be a v-DB protocol. To create the corresponding v-DB system Π^{real}, the algorithms in Π are loaded onto *devices*: e.g., physical RFID cards, NFC-enabled phones, EMV payment terminals, etc. This loading is done in accordance with the coupling presented in Def. 3.2: i.e., algorithms coupled together are loaded on the same device. For instance, in this $\Pi^{\text{real}} = ([C, PC, T], [R, W], X, \mathbb{B})$, the card algorithm C, the proximity-checking algorithm $\mathcal{PC}$ and the trusted execution environment will be loaded on the same device $[C, PC, T]$, whereas the reader algorithm $\mathcal{R}$ and validating algorithm $\mathcal{W}$ will be loaded on the same device $[R, W]$. We refer to devices that have the card algorithm C on them as ***card devices***, and devices that have the reader algorithm $\mathcal{R}$ on them as ***reader devices***.

[6] We say "some" due to the fact that v-DB protocols/systems do not attempt to formalise the underlying protocol between C and R, i.e., payment in the case of [7]; v-DB only focuses on proximity-checking aspects.

The ppt. algorithm X inside Π^{real} is also loaded onto a computational machine. When the algorithm and machine X are present in a concrete v-DB system, it is because we are faced with an *X-validated* distance-bounding system; in this case, as in Def. 3.2, note that the machine X also has algorithm $\mathcal{W}$ loaded on it. We stress that this machine onto which X (and then W) are loaded, i.e., the **X machine, is not referred to as a device**.

We consider a **ppt. *adversary* who can corrupt (card and reader) devices** (but not both the designated card and the designated reader in a given security experiment). We will define the full corruption model later. We consider that the adversary has his own **adversarial devices**, running arbitrary ppt. algorithms.

A ***party*** is an executing device (be it card, reader or adversary-owned device), or an executing X machine. Each party Y has a unique public identifier i and, as such, it is denoted Y_i.

Sessions. We allow multiple parties of the same type. Also, each party can execute multiple concurrent runs. One run of a party is called a *session*. If one execution is run on a card-device or reader-device, then it is a *card session* or a *reader session*, respectively. We write Y^i for the *i-th session of a party Y*.

Each card and reader party involved in an execution has a *status*: active or inactive. When a card or reader is inactive, it ignores all incoming messages. Initially, all are inactive. A party is only active when it is involved in one or more sessions, and becomes inactive again when this/these finish.

The chronologically-ordered list of the messages sent and received by a party in a session is called the *transcript* of the session. All sessions are attributed a unique identifier. (e.g., via the application of the pseudorandom function to the transcript). A session is *full* if its transcript contains the last message of the specification. Otherwise a session is *partial*.

The transcripts of a card-session and "corresponding" sessions of other types (reader, X) may differ, due to adversarial manipulation of messages.

If a series of partial sessions of type card, reader and (potential) X, with or without adversarial parties involved, that when put together make a partial execution of Π, then they form a *partial **macro-session***. If this leads to a full session of the party onboarded with W, then the macro-session is said to be *full*.

4.2 Physical & Communication Model.

From here on, we describe a DB-driven model that focuses mainly on the communication between card and reader devices, in a concurrent setting and in the presence of an adversary. Unlike in traditional DB model, we will also consider the communication with the X machine, which will remain honest. Also, unlike in a traditional DB model, the adversary will be able to corrupt devices in a fine-grained manner: i.e., just one specific algorithm on a device, as opposed to the whole device.

Let $\Pi = (C, \mathcal{R}, \mathcal{T}, \mathcal{PC}, \mathcal{W}, \mathbb{B})$ be a v-DB protocol, and Π^{real} be a v-DB system.

We assume that there is a **global clock**.

4.2.1 ***Computation.*** We assume that the computation of messages to send out, as well as the write and the read to memory are instantaneous.

Communications inside Π^{real} have durations, measured in units of time (e.g., seconds or fractions thereof). All messages travel at the same speed, irrespective of their length. Next, we further define these durations.

4.2.2 ***Communication Within Couplings.*** Within couplings that involve T (i.e., $[C, PC, T]$ and $[R, PC, T]$), we assume that communication happens at a constant speed. Synonymously, we consider that *to exchange a message inside a T coupling takes a* ***time-bound*** Δ: i.e., to send/receive something between (PC, T) and C, or to send/receive something between (PC, T) and R takes exactly Δ units of time.

With respect to couplings that involve W (i.e., $[R, W]$ and $[C, W]$ and $[X, W]$), we are not interested in communication times. However, we assume that these take at most a given, finite amount of time; notably, in a $[R, W]$ and $[C, W]$ and $[X, W]$ session, if this given amount of time elapses and the last response to W is not received, then Out_W is set to 0.

4.2.3 ***Communication Across Couplings.*** This means communications between card parties, reader parties, the X machine(s), as well as between these and adversarial parties. In fact, we will not be interested in the communication times with the X machine(s).

To this end, we consider that card, reader and adversary parties are positioned in a Euclidean space. The X machine is not part of this positioning.

We say that two (card, reader or adversary) parties Y_1 and Y_2 are *close* if the Euclidean distance $d(Y_1, Y_2)$ between them is at most $\mathbb{B}$: i.e., $d(Y_1, Y_2) \leq B$. Otherwise, the two parties are said to be *far apart*.

All messages exchanged between (card, reader or adversary) parties are broadcast over insecure channels and travel at a constant speed. In particular, there exists a ***time-bound*** $\mathbf{t}^{\mathbb{B}}$ such that a message from a party Y_1 reaches a party Y_2 (across couplings) within the time $t^{\mathbb{B}}$ if and only if party Y_1 is close to party Y_2.

There also exists another timing-out time-bound in which if PC inside $[PC, T]$ does not receive a response to a challenge, then it terminates unsuccessfully, i.e., $Out_{[PC,T]}$=1.

The communication with machine X is done via unicast messages and on secure channels. To partially re-iterate, we are not interested in the physics of the channel (e.g., communication speed/time) or durations of computation[7] .

4.3 Threat Model.

Adversarial Capabilities – Informal. We now present the capabilities of our adversary, informally. This informal presentation helps with the presentation in the rest of the section, whereby finally we also conclude with a formalisation of the adversary.

Let Π be a v-DB protocol.

(1). The adversary has a number of instances, at most polynomial in the security parameter, all located in the Euclidean space considered. Each instance (or *adversarial device*) implements an arbitrary ppt. algorithm.

(2). $\mathcal{A}$ can interfere with the setup of Π into Π^{real}. In this case, the adversary can corrupt the C and/or R and/or PC parts of the devices, but it cannot corrupt the T, W and X algorithms. The adversary cannot change the coupling created by the setup. For the corrupted parts, $\mathcal{A}$ can read all their material (i.e., a white-box access), but the adversary cannot modify it. Also, if a part of the coupling/device is corrupt, then then the other part does not become corrupt, yet all communications between the two parts are compromised.

(3). As usual, corruption is immutable: once corrupted, all sessions of that device are visible and can be manipulated to the attacker; we will be more specific w.r.t. coupling, in the formal parts to follow.

(4). $\mathcal{A}$ interacts with non-adversary parties by sending them messages: opening sessions with them, interfering in sessions that these parties had already started. These parties reply honestly as per Π.

(5). $\mathcal{A}$ can *move* card-parties from one location in the metric space to another. Any such move takes as much time as a message would take to travel between the two points in the Euclidean space.

(6). Adversarial instances operate as ITMs: they collaborate and communicate.

(7). The messages between adversarial devices and corrupted devices are subject to the same intra/inter-coupling communication laws as per the above in Subsection 4.2. Notably, the adversary cannot change the speed of communication of messages (i.e., make them go faster).

(8). The adversary cannot modify the global clock, except for with a negligible amount.

(9). $\mathcal{A}$ can *send unicast messages*, which can only be read by their intended target, *e.g.*, using directional antennas.

(10). $\mathcal{A}$ can *block* any message from being received by a party of his choice, irrespective of their position.

(11). $\mathcal{A}$ can *modify* messages on the fly, i.e., read and flip bits without introducing a delay to the communication.

(12). Messages sent by $\mathcal{A}$ have priority, i.e., if a bit b sent by $\mathcal{A}$ arrives to an honest party B at the same time as another bit b' sent by an honest party C, then B ignores the bit b' sent by C in preference of the bit b sent by $\mathcal{A}$.

Remarks on Our Adversarial Model. Note that our adversary is generally much aligned with "standard" distance-bounding (DB) adversary [14]. The new aspects compared to this model are as follows:

- the adversary can move parties (see point (5) above).
- the adversary can do fined grained corruption by only controlling part of a device (see control of couplings in point (2) above); this is a mix between whitebox and blackbox corruption in distance-bounding (DB) [4], and allows for the definition of new DB security properties, as we will see in Section 4.4.

Let us discuss the main restrictions of this adversary. Firstly, the adversary cannot change the coupling created by the setup of Π^{real} (see point (2) above), and that even if he corrupts one part of a coupling (say $\mathcal{PC}$ inside $[PC, T]$), then he cannot change the communication laws within the coupling (see point (7) above). In practice, this implies that the attacker cannot overclock the host of the trusted device T, i.e., the reading/writing speed of the interface to/from T stays as prescribed by the specifier of T. Secondly, as per point (8) above, the adversary cannot control the global clock; this

[7] By this we mean that we will not look at how long a message computation takes inside the machine X. However, we do not mean that we allow X to be in any complexity class; in fact, from a complexity viewpoint, the X machine runs a ppt. algorithm

is in line with "standard" distance-bounding (DB) models [14], but it also captures the fact that there is root of trust in the presence of T; implicitly, we consider that T provides this trusted, tamper-resistant clock[8].

Challenger. To simulate the honest executions of a v-DB system Π^{real} as well as its interactions with the adversary we introduce, as usual, a *challenger*.

(1). The challenger Ch is aware of the global clock.

(2). The challenger Ch keeps a list $\mathcal{P}$ of all parties in the system, indexed by their id. Each entry in $\mathcal{P}$ also contains their corruption status (corrupt/honest), and –for card and reader parties– the entry also contains their location[9]. If parties/devices are coupled (as in $[C, PC, T]$), then –in the list– they appear with the entire coupling. This is because it is the challenger Ch who ran the setup in Π^{real}. Also, Ch deals with all adversarial actions via a set of oracles presented later; as such, challenger Ch knows if a given party has been corrupted by $\mathcal{A}$ and his list $\mathcal{P}$ is kept up-to-date accordingly.

(3). The challenger Ch keeps track of every session, opened by every party in a list called *Sess*. This list is index by the unique session identifier, and it registers the time the session started, if it is a card session, a reader session, etc., as well as the up-to-date status of a session: i.e., finished or running, and a transcript of the session.

(4). The challenger Ch keeps a list *Sends* of timed, sent messages. This contains: the id of the session (of the sender party) to which this is message belongs to, the sender party, the aimed receiver party (which is optional), the message, and the time of send. Recall that most messages are sent in broadcast mode, and only the adversary can send messages in unicast mode; so, the latter is the only case in which is there is an aimed receiver.

(5). The challenger Ch keeps a list *Reads* of read messages at given times. This contains: the id of the session (of the reading party) in which this message is being read, the (apparent) sender party, the (real) sender party, the receiver party, the message, the time of the receipt. We will formalise this later. However, we mention the following two important **time-keeping aspects** here. Firstly, if the "read" is from/to a sender and receiver across different couplings, then an entry in this list is possible only if the message appears in the sent-messages list *Sends* and if the message had the time to travel from the sender to the receiver. I.e., $d(sender, receiver) \leq (t_{sent} - current\_time) \times c$, where the challenger Ch finds the locations of $sender$, $receiver$ in the $\mathcal{P}$ list , the time t_{sent} in the *Sends* list, the $current\_time$ by using the global clock, and c is the speed of messages. If this inequality holds, then the time of receipt inside *Reads* is recorded as the $current\_time$. Second, if the "read" is from/to a sender and receiver inside the same coupling, then an entry in this list is possible only if the message appears in the sent-messages list *Sends* and the fixed amount of time passed since the message was sent. I.e., the challenger Ch finds the locations of $sender$ and $receiver$ in the $\mathcal{P}$ list , the time t_{sent} in the *Sends* list, the $current\_time$ by using the global clock, and $current\_time - t_{sent} = \delta$.

The points above show that the challenger Ch is an arbiter for the setup of the system, honest and corrupt behaviours, and the communication rules in our model. Specifically, w.r.t. point (5) above, the challenger Ch uses his "communication log" kept in the lists $\mathcal{P}$, *Sess*, *Sends* and *Reads*, so that he does not allow the communication rules expressed in our model to be broken: communication across couplings is proportional to distance, and communication inside couplings takes a fixed amount of time.

Oracle-based Security Model. As we will see now, the adversary and the challenger's aforementioned behaviours are formalised (as per the usual) through an interaction modelled via *oracles*. In this sense, the adversary will call oracles to the challenger in order to simulate real executions between honest and corrupted parties.

The set of oracles below is denoted as **v-DB-Orcls**.

- **init**$([C, \mathcal{R}, \mathcal{T}, \mathcal{PC}, \mathcal{W}, \mathcal{X}, \mathcal{A}])$

  This oracle allows the adversary to initiate a session between a meaningful subset of the parties listed as potential inputs. The challenger Ch checks the format of the call, and only meaningful calls are allowed. E.g., On the one hand, if a card-party C_i is not coupled, then the adversary can call $init(C_i)$. On the other hand, if a card-party C_i is coupled, then the adversary can only call $init(C_i, \mathcal{PC}, \mathcal{T})$, where $\mathcal{PC}$ and $\mathcal{T}$ are the parties coupled with C_i.

  The adversary can also call $init(C, \mathcal{R}, \mathcal{T}, \mathcal{PC})$ with the correct coupling included in the call (i.e., $init([C, \mathcal{PC}, \mathcal{T}], \mathcal{R})$ or $init([\mathcal{R}, \mathcal{PC}, \mathcal{T}], C)$, otherwise the call is rejected). In this case, it means that the parties in the call will start sessions in which they communicate together. To this end, the challenger Ch records in the *Sess* list different times for different parties depending on their location. I.e., the initiator of the session is logged inside *Sess* to have started the session at the current time t, the responder in the session at time $t + distance(initiator, responder)$. Note that the challenger Ch has the location of all parties in $\mathcal{P}$ and knows the protocols, so it can determine if e.g., a card party or a reader party is the initiator or responder of the sessions, respectively. If $\mathcal{W}$ is used inside the call, then the time of the recorded open session for $\mathcal{W}$ is the same as the party coupled with $\mathcal{PC}, \mathcal{T}$.

  Note that some parties can be corrupted (see oracle $corrupt(\ldots)$ later on) and, as such, this session-creating is also over corrupt cards and *PC* parties.

  If $\mathcal{A}$ is used inside the call, then the adversary "starts" a session with one/some of the other parties in the call.

  The challenger Ch will record the opened session(s) accordingly in the *Sess* list. If all parties in a Π^{real} are given at input, then different session ids are created for each, and a macro-session id *mid* is created for the set. The *mid*s and macro-session are also recorded in *Sess* (i.e., each *sid* can be linked to a *mid* and its details) inside *Sess*.

  The ID *sid* of the opened session(s) per each party is given as output (to $\mathcal{A}$).
- **send**$([\mathbf{sid}, \mathbf{S}, \mathbf{R}, \mathbf{m}])$

  This oracle generally denotes the sending of a message m inside session *sid* from sender S aimed to the receiver R. As

[8]This is in line with the specification for Trusted Platform Modules (TPMs) by their standardisation body – TCG (Trusted Computing Group).

[9]We can also trade space-complexity for time efficiency and also keep dedicated lists of corrupted parties, dedicated lists for locations, etc. This type of trade-off can be considered on all Ch's lists; when implementing a mechanisation of this model in a cryptographic prover, such multiple lists with redundancy are beneficial for faster proofs.

we explain next, only S is required, the rest of the parameters are optional.

All checks mentioned below are done by the challenger Ch via the lists $\mathcal{P}$ and *Sess*.

If no *sid* is specified, then the message is sent to all sessions of the party passed under the R parameter. If S is equal to $\mathcal{A}$, then m must be specified, meaning that the adversary is sending a message m. If a *sid* is specified, then this session id *sid* must exist in *Sess*. If S is equal to $\mathcal{A}$, plus the receiver R and *sid* are specified, then R must have a session *sid* open (which the challenger Ch checks in *Sess*), and only the party R can read this message (i.e., $\mathcal{A}$ is sending a message in unicast mode). Contrarily, if $send([S, \cdot, m])$ with $S \neq \mathcal{A}$ is called, then this messaging emulates broadcast, i.e., all possible parties who as per Π can receive m will be forwarded this message.

The adversary can call this oracle with non-adversarial parties as parameters, in to order get a simulation of communications between honest or corrupt devices. I.e., in the template of the oracle $send([sid, S, R, m])$, we have that $S, R \in \{C, \mathcal{R}, (\mathcal{PC}, \mathcal{T}), (\mathcal{W}, \mathcal{Y}) \mid Y \in \{\}\}$ in such a way that compliance with the protocol specification, the coupling and the session ids is maintained. That is to say, the Ch makes the necessary checks: e.g., if both an S and an R are passed in the oracle, then they communicate as per the protocol, if the *sid* is specified then the sender S has such a session in *Sess*, etc.

If all checks pass, then the message will become part of this session's transcript (inside the list *Sess*) and the sending will be recorded in the *Sends* list (which was described above). If the checks do not pass, the challenger discards the call.

- **read([sid, S, R, m])**

  This oracle has two inputs that are obligatory: R, and either *sid* or m. This means that party R should read a message in session *sid*, or read the message m. If all are specified, then clearly it means that party R should read message m in *sid* and consider it as coming from party S. Any subset of inputs is implicitly explained by this. The challenger Ch makes the necessary checks by looking in *Sends*: i.e., (1)seeing if S or $\mathcal{A}$ have sent the message m; (2) if in session *sid*, R is at the step to read m; (3) if the difference between the time t_{sent} when m was sent by (the honest or adversarial) S , and the current time t is large enough for the message to have travelled from the sender to the receiver R (i.e., $(t - t_{sent}) * c \geq distance(S, R)$, where c is the speed of messages).

  If all checks pass, the message will become part of this session's transcript (inside the list *Sess*) and the reading will be recorded in the *Reads* list (which was described above). If the checks do not pass, the challenger discards the call.

  Note. In this formalisation, no two messages can be sent at the same time. Yet, $read([sid, S, R, m_1])$ and $read([sid, S, R, m_2])$ could be enabled at the same time for a receiver R. In this case, if one of the two messages are sent by the adversary or a corrupt party, and the other message by an honest party, then the one sent by the adversary has priority, i.e., the challenger will record the read of the adversarial message first[10].
- **corrupt(E)**

  The adversary calls this oracle to corrupt parties/algorithms; E can be a party of type card, type reader, or the PC algorithm coupled inside a reader or card party. In the latter case, the input has to specify the coupling.

  The challenger looks in its list $\mathcal{P}$ and if the party exists, it changes its status to "corrupt". Otherwise, if the input is malformed or the checks fail, the challenger discards the call.
- **term(E)**

  This oracle terminates all the running sessions of the party E. This gets recorded in the list $\mathcal{P}$ (where the party E is made inactive), and in the lists *Sess* where its sessions are set to "finished".
- **move(E, loc)**

  This oracle moves a party E from its location to another point *loc* in the metric space. The challenger Ch checks in $\mathcal{P}$ that the party E is of type card and that it is corrupt. In this case, the challenger calculates the difference between E's current location loc_1 (found in $\mathcal{P}$) and *loc*, and in proportional amount of time, it updates[11] its records in $\mathcal{P}$ to stipulate that E is at location *loc*.
- **check_prox(sid)**

  The challenger checks that *sid* is a valid session for $[PC, T]$. If so, it takes its transcript from *Sess* and passes it to $[PC, T]$. From this, the attacker is given: $Out_{[PC,T]}$ and $\tau_{[PC,T]}$, i.e., the public output and the private output of $[PC, T]$, as well as the id of the card C and reader R who were measured by $[PC, T]$.

  In *Sess*, the entry for *sid* is marked as "finished".
- **validate_prox(sid)**

  The challenger checks that *sid* is a valid session id for W. If so, it takes its transcript from *Sess* and passes it to the coupling $[W, X]$. From this, the attacker is given: Out_W and τ_W, i.e., the public output and the private output of W, as well as the id of the card C and reader R who were measured by $[PC, T]$ and revalidated by W.

  In *Sess*, the entry for *sid* is marked as "finished".

Other Actions by the Challenger. The list *Sess* of sessions is maintained by the challenger Ch also out-of-bound with the oracles above. Namely, there is a time-keeping aspect that we mentioned in Subsection 4.2.2 and Subsection 4.2.3: i.e., if certain messages in a PC or W-linked session take too long to arrive, then that session is marked as "finished" and the transcript contains Out_{PC}=0, Out_W=0 inside the list Sess.

***Remarks on the* v-DB *Security Model*.** With respect to proximity-checking, the model introduced above is arguably standard. Notably, timing and distance-related aspects are explicit as in [14], with the addition that it extends this to operate over couplings (intra and across couplings). Of course, this is inherent to the fact that we lift DB protocols to validated DB protocols; to this end, we also

[10]Note that this prioritisation mechanism also encapsulates blocking of any messages by the adversary $\mathcal{A}$.

[11]Note that "moves" of parties are also subject to same laws of physics as messages, thus not hindering the latter.

incorporate the management of the communication (incl. timed aspects thereof) with $[W, Y]$ with $Y \in \{C, R, X\}$. What this model has that was notably not present in previous DB models is the fact that the attacker can move (card) parties. Last but not least, our model enjoys a precise session management, which is rooted primarily on the fact that there are multiple parties involved in the primitive of v-DB (compared to a standard "DB" primitive).

4.4 Security Properties for Validated Distance Bounding

In this section, we define the correctness and security of v − DB systems.

4.4.1 ***Correctness of v-DB Protocols.*** Now, in Def. 4.1, we define what it means for a v-DB system to terminate and to run correctly. Concretely, firstly, Def. 4.1 says that the no matter which algorithms run on the card and reader, eventually PC and W terminate[12]. Secondly, Def. 4.1 says that if an arbitrarily fixed, honest card is close to an arbitrarily fixed honest reader, coupled (in any way) with an honest proximity-checking algorithm, then the proximity-checking and the validating algorithms finish successfully with probability p.

Definition 4.1. **Correctness of Validated Distance Bounding**. Let $\Pi = (C, \mathcal{R}, \mathcal{T}, \mathcal{PC}, \mathcal{W}, \mathbb{B})$ be a v-DB protocol, and $\Pi^{\text{real}} = (C, PC, T, R, W, X, \mathbb{B})$ be a validated DB system. We say that Π and Π^{real} are *terminating* and *complete validated distance-bounding* if the following holds, respectively:

- **Termination**:
  $(\forall s)(\forall C'$ unbounded$)(\forall R'$ unbounded$)$,
  for any **init**$[C', R', (PC, T), W, X]$ call yielding a macro-session mid, it is the case that (PC, T) and W halt in $poly(s)$ computational steps and $finished$ is recorded in $Sess$ for the sessions of $[PC, T]$ and W corresponding to the macro-session mid, where s is the security parameter;
- p-**Completeness**: $(\forall s)$, if $distance(C, R) \leq \mathbb{B}$, we have the following:
  for any **init**$[C, R, (PC, T), W, X]$ call yielding a macro-session mid, with sid and sid' being the respective sessions of $[PC, T]$ and W corresponding to the macro-session mid, we have that

$$\Pr_{r_C, r_R, r_{PC}, r_T, r_W}\left[(Out^{sid}_{(\mathcal{PC},\mathcal{T})} = 1 \wedge Out^{sid'}_{\mathcal{W}(\tau_{(\mathcal{PC},\mathcal{T})})} = 1) : \begin{array}{l} mid \text{ running} \\ sid, sid' \in mid \end{array}\right] \geq p,$$

for any arbitrarily fixed C, R, PC uncorrupted, for any arbitrarily fixed T, W, X, with $r_C, r_R, r_{PC}, r_T, r_W$ being the random coins in the macro-session mid of the algorithms mentioned in the indices.

In practice, p the parameter for correctness (in Def. 4.1) needs to be tuned with the parameters that will define our security (see such tuning in e.g. [5]). However, for security to be meaningful we require that p is always overwhelming in the security parameter.

4.4.2 ***Security of v-DB Protocols.*** We give our definition of security for v-DB systems in terms of a game. Roughly speaking, in this game, the attacker can play with multiple cards, multiple readers, and validating bodies, in a given coupling: either they are all card-coupled or all reader-coupled. There is a target card and a target reader. The attacker can open multiple sessions with each such party. He can corrupt cards, readers and PC (at any point), and he can move (corrupted) cards; some restrictions apply over the different phases of the game. Indeed, the game has two phases: (a) a learning phase in which the target parties may be in mutual proximity but are not corrupt and nor is the PC on board one of them. (b) an attack phase – in which the target parties are far away and the one that has PC on board can be corrupted, as can its on-board PC. As the corrupted PC is made to output 1 (i.e., lie about the proximity of the target parties), the attacker wins if he manages to make the validating party $[W, X]$ also output 1. In other words, the attacker wins if it is able to manipulate a number of parties, possibly in a concurrent setting, in order to make the validating algorithm be fooled w.r.t. to a lie by the corrupt PC algorithm, that W is supposed to audit.

In fact this game-based definition can be split in two: (a) the case where the PC is on-board the "target" reader; (b) the case where the PC is on-board the "target" card. In each case, indeed, it yields different security properties. In the second case, the attacker would collude with a far-away PC-coupled card to mount a distance-fraud that is undetected by the validating algorithm. In the first case, the attacker would collude with a PC-coupled reader so that they together mount a strong relay attack attack that is undetected by the validating algorithm.

In our case, we define the case in a generic game, i.e., comprising both of the above cases together. Then, we simply fork the two cases for the security definitions. This generic game is captured in Def. 4.2 by the notion of $(\ell, z, n, q_C, q_R, q_T, \texttt{type})$-**v-DB** ***experiment***, where type can be "***card-coupled***" or "***reader-coupled***", ℓ are number of card-parties, z are number of reader-parties, n are number of $[PC, T]$ on-board algorithms and X-machines, q_C, q_R, q_T are number of sessions of type card, reader, and T, respectively.

Definition 4.2. $(\ell, \mathbf{z}, \mathbf{n}, \mathbf{q_C}, \mathbf{q_R}, \mathbf{q_T}, \texttt{type})$-**v-DB Experiment**. Let Π an v-DB protocol. For any security parameter s, an $(\ell, \mathbf{z}, \mathbf{n}, \mathbf{q_C}, \mathbf{q_R}, \mathbf{q_T}, \texttt{type})$-**v-DB** ***experiment*** is an interaction between the Challenger and Adversary as follows:

(1) The Challenger setups a v-DB system with:
- $\ell > m$ card-parties $C_1(\cdot), \ldots, C_m(\cdot), \ldots, C_\ell(\cdots)$ and a card-party $\overline{C(\cdot)}$;
- z reader-parties $R_1(\cdot), \ldots, R_p(\cdot), \ldots, R_z(\cdot)$ and a reader party denoted $\overline{R(\cdot)}$;
- n on-board algorithms $[PC_1, T_1], \ldots, [PC_n, T_n]$ and one other on-board algorithm denoted $[\overline{PC}, \overline{T}]$;
- a n machines $X_1, \ldots, X_n$ and one X-machine denoted $\overline{X}$;
- apart from the cryptographic material used to authenticate $\overline{C}$ to $\overline{R}$, the other card-parties C_is and reader-parties R_js can respectively have the same cryptographic material[13] used in the authentication part of the v-DB protocol Π.

[12] In our model, this is in part guaranteed implicitly by the conditions mentioned in Subsection 4.2.2 and Subsection 4.2.3 and incorporated formally in the challenger's behaviours.

[13] Whilst the devices are different, with different device ids, it is the case that e.g., two cards are have the same cryptographic credentials. This, together with the corruption powers described below, entails a powerful attacker model. I.e., in the learning phase of the experiment, the attacker can corrupt a card-party C_i running on cryptographic material x, making it behave as he pleases and potentially extracting x out of it. This means that the attacker gets the capability to, e.g., sign, as a legitimate card based on x. Moreover, in the attack phase of the experiment, another card-party C_j with the same cryptographic material x can also be present (alongside the "learned" adversary), only that this time we do restrict that C_j per se is un-corrupted.

where values m, ℓ, z, p, n are in $poly(s)$, and all $[PC, T]$s are either coupled with the cards, or all with the readers[14] (i.e., n=m or n=z). If the coupling of $[PC, T]$s is with the cards, then the type of the experiment is "card-coupled"; we call this a ***card-coupled experiment***.

If the coupling of $[PC, T]$s is with the readers, then the type of the experiment is "reader-coupled"; we call this a ***reader-coupled experiment***.

(a) If the experiment is a ***card-coupled experiment***, then the adversary can call **corrupt**(...) at any point, but the only on card-parties and PC algorithms. And, not all card/PC parties can be corrupted in any phase of the experiment.

(b) If the experiment is a ***reader-coupled experiment***, then the adversary can call **corrupt**(...) at any point, but only on reader-parties and PC algorithms. And, not all reader/PC parties can be corrupted in any phase of the experiment.

(c) Before any call by the adversary, the Challenger populates the $\mathcal{P}$ list as follows:
– for parties $C_1(\cdot), \ldots, C_m(\cdot)$ and parties $R_1(\cdot), \ldots, R_p(\cdot)$, the positions are chosen arbitrarily in the Euclidean space;
– for card-parties $C_{m+1}(\cdot), \ldots, C_\ell(\cdot), \overline{C(\cdot)}$, the reader-parties $R_p(\cdot), \ldots, R_z(\cdot), \overline{R(\cdot)}$, the positions are chosen such that the distance between any of these card and reader parties are bigger than the bound $\mathbb{B}$.

(2) In the **learning phase** of the experiment, the adversary is allowed access to the first m out of ℓ card-parties and $\overline{C(\cdot)}$, the first p reader-parties and $\overline{R(\cdot)}$, the $[PC, T]$ algorithms and X machines fitting the coupling made with these card and reader devices.

(3) The Challenger will reveal the id of $\overline{C(\cdot)}$ and $\overline{R(\cdot)}$.
(a) The entire set **v-DB-Orcls** of oracles defined above is available to the adversary, restricted as per the below.
–The adversary cannot call **corrupt** on $\overline{C(\cdot)}$, $\overline{R(\cdot)}$ or $\overline{PC(\cdot)}$.
–The adversary is allowed a polynomial number of queries of each type of oracle.
–The experiment starts as soon as at least one macro-session is running as a result of a call of the type **init**(C, R, PC, T, W, X), with parameters from the set of parties available in this phase of the game. We consider that the Challenger makes this call, for some arbitrarily picked parties in this stage.
(b) After at least one X-session is marked as "finished" in the list $Sess$, the Challenger can stop the learning phase. In any case, the Challenger stops the learning phase after a polynomial number of X sessions are marked as "finished" in the list $Sess$. This stopping by the Challenger means that all sessions are terminated.

(4) In the **attack phase** of the experiment, the adversary is allowed access to $C_{m+1}(\cdot), \ldots, C_\ell(\cdot), \overline{C(\cdot)}$, the reader-parties $R_p(\cdot), \ldots, R_z(\cdot), \overline{R(\cdot)}$, th $[PC, T]$ algorithms and X machines fitting the coupling made with these card/reader devices. The adversary is allowed access to the set **v-DB-Orcls** of oracles, as follows.

(5) The Challenger will reveal the id of $\overline{C(\cdot)}$ and $\overline{R(\cdot)}$.
(a) If the experiment is **card-coupled**:
–The Challenger will itself call **corrupt** on $\overline{PC}$ on board $\overline{C(\cdot)}$
–The Challenger will call **init**$([\overline{C(\cdot)}, \overline{PC(\cdot)}, \overline{T(\cdot)}], \overline{R(\cdot)}, \overline{W}, \overline{X})$; this produces a macro-session mid. The challenger gives the necessary handles of this to $\mathcal{A}$.
–The adversary can also call **corrupt** on $C_{m+1}(\cdot), \ldots, \overline{C(\cdot)}$ and their $\overline{PC}$s. The adversary cannot call **corrupt** on $\overline{R(\cdot)}$.
(b) If the experiment is **reader-coupled**:
–The Challenger will itself call **corrupt** on $\overline{PC}$ on board $\overline{R(\cdot)}$.
–The Challenger will call **init**$([\overline{C(\cdot)}, \overline{C(\cdot)}, \overline{T(\cdot)}], \overline{R(\cdot)}, [\overline{W}, \overline{X}])$; this produces a macro-session mid. The challenger gives the necessary handles of this to $\mathcal{A}$.
–The adversary can also call **corrupt** on $R_p(\cdot), \ldots, R_z(\cdot), \overline{R(\cdot)}$ and their $\overline{PC}$s. The adversary cannot call **corrupt** on $\overline{C(\cdot)}$.
(c) The adversary cannot call **move**$(\overline{C(\cdot)})$.

(6) The adversary can make q_C, q_R, q_T queries to **init**$(C, \ldots)$, **init**$(\ldots, R, \ldots)$, **init**$(\ldots, [PC, T], \ldots)$ in total. Of the other types of oracle, the adversary can make an unspecified by polynomial number of calls.

(7) The Challenger lets the experiment continue until $\overline{PC}$ and X finish their respective session sid and sid' inside the macro-session session mid, or until the adversary's allowed number of queries is reached.

(8) The experiment continues if $Out^{sid}_{(\overline{PC}, \overline{T})} = 1$ (since $\overline{PC}$ is corrupted, $\mathcal{A}$ gets all details of $\overline{PC}$'s session and, to finish the game, will produce this output.)

(9) The adversary wins if $Out^{sid'}(W(\tau_{((\overline{PC}, \overline{T}))}))= 1$.

(10) The advantage of the adversary is the $\Pr\left[Out^{sid'}_{W(\tau_{((\overline{PC}, \overline{T}))})} = 1\right]$, taken over all random coins in parties included in the attack phase.

Definition 4.3. **Strong Relaying Security**. For a given v-DB protocol Π, the $(\ell, \mathbf{z}, \mathbf{n}, \mathbf{q_C}, \mathbf{q_R}, \mathbf{q_T}, reader-coupled)$-**v-DB experiment** gives the ***game for strong relaying over*** Π of the same parameters.
If the advantage of the adversary is negligible is this game, then we say that Π is secure w.r.t. strong relaying.

In this case, $\mathcal{A}$ is a strong man-in-the-middle who controls the $\overline{PC}$ algorithm onboard a reader $\overline{R}$, and may control $\overline{R}$ itself. It attempts the make the attester W accept a transcript when the card $\overline{C}$ was far from the reader, when the $\overline{PC}$ algorithm already lied about this fact (i.e, $Out^{sid}_{(\overline{PC}, \overline{T})} = 1$); that is, the adversary and the $\overline{PC}$ algorithm (maybe alongside the reader $\overline{R}$ algorithm) try collude to make it look that $\overline{C}$ was close to the reader $\overline{R}$ when it was not. Moreover, this adversary, during the attack, has access to cards other than $\overline{C}$ which it can move around, and to readers other than $\overline{R}$. Also, it

[14] In order words, there is no mixing of card-coupling and reader-coupling in the system.

took part in a learning phase, in which is observed and interfered with several runs of the protocol in a concurrent setting, yet in this learning phase $\overline{PC}$ and $\overline{R}$ were honest.

Definition 4.4. **Strong Distance-Fraud Security**. For a given v-DB protocol Π, the $(\ell, \mathbf{z}, \mathbf{n}, \mathbf{q_C}, \mathbf{q_R}, \mathbf{q_T}, card-coupled)$-**v-DB experiment** gives the ***game for strong distance-fraud over*** Π of the same parameters.
If the advantage of the adversary is negligible is this game, then we say that Π is secure w.r.t. strong distance-fraud.

In this case, $\mathcal{A}$ is a strong man-in-the-middle who controls the $\overline{PC}$ algorithm onboard a card $\overline{C}$, and may control $\overline{C}$ itself. It attempts the make the attester W accept a transcript when the card $\overline{C}$ was far from the reader, but $\overline{PC}$ algorithm lied about this fact (i.e, $Out^{sid}_{(\overline{PC},\overline{T})} = 1$). In other words, the attacker corrupts a $\overline{PC}$-coupled proximity-checking card $\overline{C}$ inside a v-DB protocol and try to use this card to mount a distance-fraud that the attester would not catch. The rest of the aspects (i.e., learning phases, etc.) stay the same as in the game for strong relaying, only that readers are uniformly replaced with cards.

4.5 Security Statements

In this section, we prove that PayBCR and PayCCR, introduced in [7], and indeed secure w.r.t. strong relaying.

Theorem 4.5. **PayBCR*'s Security w.r.t. Strong Relaying.*** *Consider the* $(\ell, z, n, q'_C, q'_R, q'_T, reader-coupled)$-v-DB *experiment in the strong relaying game for* PayBCR.

If σ_1, σ_2 *are signatures unforgeable w.r.t. selective unforgeability [3] (i.e., SUF-unforgeable) and AC is produced with a MAC that resists existential forgeries, then* PayBCR *is secure with respect to strong relaying. Concretely, the advantage of the adversary is:* $\frac{q_R^2}{2^{|N_R|}} + \frac{q_T^2}{2^{|\sigma'_1|}} + \frac{2{q_C}^2}{2^{|N_C|}}$, *where* $q_C = q'_C \cdot l$, $q_R = q'_R \cdot z$, $q_T = q'_T \cdot n$ *and* $|\cdot|$ *denotes the bit-length of a protocol message.*

Theorem 4.6. **PayCCR*'s Security w.r.t. Strong Relaying.*** *Consider the* $(\ell, z, n, q'_C, q'_R, q'_T, reader-coupled)$-v-DB *experiment in the strong relaying game for* PayCCR.

If σ_1, σ_2 *are signatures unforgeable w.r.t. selective unforgeability (i.e., SUF-unforgeable), then* PayCCR *is secure with with respect to strong relaying. Concretely, the advantage of the adversary is:* $\frac{q_R^2}{2^{|N_R|}} + \frac{q_T^2}{2^{|\sigma_1|}} + \frac{2{q_C}^2}{2^{|N_C|}}$, *where* $q_C = q'_C \cdot l$, $q_R = q'_R \cdot z$ *and* $q_T = q'_T \cdot n$ *and* $|\cdot|$ *denotes the bit-length of a protocol message.*

The proofs of these theorems are found in Appendix A.

Note that because in PayCCR, it is the AC-producing card that has the validating algorithm on board (as opposed to PayBCR, where this AC needs to be sent by the card to the validating bank), the requirements for PayCCR's security w.r.t. strong relaying are weaker than those of PayBCR. I.e., the security of AC plays no role in PayCCR's security w.r.t. strong relaying. This imbalance will be inverted if we look at auditing authentication properties as well. In this case, it is will be PayBCR that achieves better guarantees w.r.t. strong, collusive attacks against authentication. This is discussed further in Section 5.

5 OTHER DISCUSSIONS

5.1 v-DB, PayBCR, PayCCR: Further Security Discussions

When looking at v-DB , it is important to note that we defined security for only the properties of strong relaying and strong distance-fraud. As such, we may wish to look at a primitive that can give assurances of both proximity checking and authentication. This would be comprised by having a "validating" party W to re-check the proximity measurement and "auditing" party, say $\mathcal{F}$, to re-check the authentication part. This split would clearly also open for other coupling options than in v-DB. For instance, PayCCR has the validator for the proximity checking with one party (that is, on the card side) and the auditor for authentication with another party (that is, on the bank side). Contrarily, with PayBCR , the auditor for authentication and the validator for proximity checking validation occur both with the same party (i.e., on the bank's side).

We now define such a protocol.

Definition 5.1. **Validated & Audited Distance-Bounding Protocols**. A *validated and audited distance-bounding (*v-ADB*) protocol* is a tuple $\Pi = (\mathcal{C}, \mathcal{R}, \mathcal{T}, \mathcal{PC}, \mathcal{W}, \mathcal{F}, \mathbb{B})$, where $\mathbb{B}$ denotes the distance bound and $\mathcal{C}, \mathcal{R}, \mathcal{T}, \mathcal{PC}, \mathcal{W}, \mathcal{F}$ are ppt. algorithms as follows:

- $\mathcal{C}$ is the card algorithm and $\mathcal{R}$ is the reader algorithm in an unilateral authentication protocol where $\mathcal{C}$ authenticates to $\mathcal{R}$;
- $\mathcal{T}$ is a tamper-resistant trusted execution environment;
- the tuple $(\mathcal{PC}, \mathcal{T})$ form a proximity-checking functionality: $(\mathcal{PC}, \mathcal{T})$ checks that $dist(\mathcal{C}, \mathcal{R}) \leq \mathbb{B}$;
- tuple $(\mathcal{PC}, \mathcal{T})$ is directly used by one authentication party: be it by $\mathcal{C}$ or by $\mathcal{R}$;
- $\mathcal{R}$ and $(\mathcal{PC}, \mathcal{T})$ respectively have public outputs Out_R and $Out_{(\mathcal{PC},\mathcal{T})}$ in $\{0, 1\}$ (success/failure of the authentication and proximity-checking respectively), as well as private outputs denoting their transcripts, denoted $\tau_{\mathcal{R}}$ and, resp., $\tau_{\mathcal{PC}}$;
- $\mathcal{W}$ is the proximity-validating algorithm: given the private output of $(\mathcal{PC}, \mathcal{T})$, the algorithm $\mathcal{W}$ checks the correctness of public output of $(\mathcal{PC}, \mathcal{T})$.
- $\mathcal{F}$ is the authentication-auditing algorithm: given the private output of $\mathcal{R}$, the algorithm $\mathcal{F}$ checks the correctness of the public output of $\mathcal{R}$

Security in this setting can be defined in various ways, some stronger than others. E.g., we can say that a v-ADB protocol Π has **strong MiM v-ADB-security** if for all cards $C(x)$ far away from potentially malicious readers $R(y)$, their authentication fails to be audited by F with input from $R(y)$ and W fails to validated $C(x)$ was close to $R(y)$ (even if the output of corrupted PC is 1). Clearly, similarly we can defined **strong v-ADB-distance-fraud**, but where the card may be malicious instead of the reader. Of course, this would need to be cast in threat model akin to ours but where more authentication-forging powers are given to the adversary. We leave this for future work.

Collusive relaying, as informally defined in [7], is a property that is stronger then the property of strong relaying defined for v − DB, but weaker than **strong MiM v-ADB-security**. It requires intuitively that if F audits successfully an authentication, then this

authentication is for a card that was close to a reader, even if the reader and the *PC* algorithm may be malicious.

v-ADB ***Security: Protocol Comparisons.*** As we explained already, Def 5.1 makes a difference for PayCCR but not for PayBCR. Concretely, in PayBCR $\mathcal{F}$ and $\mathcal{W}$ are on the same device (i.e., the bank), but in PayCCR, $\mathcal{F}$ is one party (i.e., bank) and $\mathcal{W}$ is another party (i.e., the card). In turn, this intuitively makes PayCCR weaker than PayCCR. More specifically, assume a card C that is far-away from a reader R. In PayCCR, we have that $[PC, T]$ is coupled with the reader, and W coupled with the C. Even if $\mathcal{W}$ is coupled with the card, the AC produced by the card will not contain the failed checks by $\mathcal{W}$. (The latter is because, in PayCCR, the AC is kept unchanged compared to RRP; also, the AC is to go to $\mathcal{F}$, and –as such– the AC need not contain proximity-checking details but only authentication details). This *AC* will be sent to the reader who will send it to the bank. So, the bank authenticates a far-away card, even if the auditing $\mathcal{W}$ will report the proximity-checking failing. This is an attack against strong MiM v-ADB-security informally exhibited onto PayCCR. Because, in PayBCR, $\mathcal{F}$ and $\mathcal{W}$ are kept on the same device, this vulnerability does not occur.

6 RELATED WORK

Models for "standard DB". Most distance-bounding protocols have been analysed without a formal approach. From 2013 onwards, efforts have been made on proving security for distance-bounding [11, 13, 14]. The first formalism in this direction was put forward by Dürholz *et al* [11]. The authors formalise the impossibility of illegitimate yet sufficiently fast round-trip communications using a session-based model, and specifically the notion of *tainted sessions*; to encode timing-restrictions, tainted sessions only allow certain flows of communication. Then, a protocol is said to be secure if no adversary executing it with tainted sessions can violate its security properties. The model comprises a formalisation of all the classical DB frauds and provides several (partial) security proofs for some protocols [11]. In [14], the authors provide a rather general, ITM-based model that captures the notion of concurrency (i.e., allowing adversaries to interact with many provers and verifiers, sometimes with the same keys). Here, the notion of timing is explicit, the round trip time is simply the difference between two times. The notion of distance is also defined in a similar way. All parties are located in Euclidian space with a distance between them defined in the usual manner. As such the explicit measure of time can be used to estimate the distance between two parties to compare with a distance bound.

In this paper, we use a mix of both of these ideas. We use oracles in the sense of Dürholz *et al* [13], but we also make use of explicit timing and distances as in [14].

Security Models for Distance-Bounding Augmented with Hardware. In [15], a "three-algorithm symmetric DB protocol" is defined as a tuple of (K, V, P, B, H), where K is the key generation algorithm, H is the hardware algorithm, V is the verifying algorithm, P is the prover algorithm and B is the distance bound. However, the trusted hardware H is always attached to (or in our terms, "coupled with") the Prover, whereas in our model the secure element is coupled with the proximity-checking algorithm *PC*, and this *PC* can in turn be coupled with either the card (prover) or the reader (verifier). This makes our model more versatile and expressive. Also, ultimately, the aims of [15] are very different to ours: they wish to protect against a specific attack in DB, called terrorist-fraud; the latter has no substantial relation to our notions of strong relaying and strong distance-fraud.

Security Models for Relay-Protected Contactless Payments. In [16], Kilinç and Vaudenay introduce a model for contactless payments with relay-protection. This model is distinct to the model herein, in ways others that the one herein is generic and the one in [16] is specific to payments. The major difference is that the reader is always trusted.

7 CONCLUSIONS

In this paper, we set to answer the question of what it would formally mean to catch if RTT-measuring parties (readers, cards, or others) cheat and collude with proximity-based attackers (i.e., relayers or other types). To this end, we gave a new distance-bounding primitive (*validated distance-bounding*) and two new security notions: *strong relaying* and *strong distance-fraud*. We also provided a formal model that, for the first time in distance-bounding, caters for dishonest RTT-measurers. In this model, we proved that the new contactless payments in [7], PayBCR and PayCCR attain security w.r.t. strong relaying. Finally, we define one other primitive (validated and audited distance-bounding) which, in fact, emulates more closely the PayCCR protocol; this is because, contrary to the line introducing them, we note that PayBCR and PayCCR in fact differ in construction and security guarantees that go past relaying and into authentication. In future work, we plan to study further formal security (now just sketched) alongside validated and audited distance-bounding.

Acknowledgments. The authors acknowledge the support of the NCSC-funded "TimeTrust" and "PayPhy" projects.

REFERENCES

[1] G. Avoine, M. Bingöl, I. Boureanu, S. Čapkun, G. Hancke, S. Kardaş, C. Kim, C. Lauradoux, B. Martin, J. Munilla, A. Peinado, K. Rasmussen, D. Singelée, A. Tchamkerten, R. Trujillo Rasua, and S. Vaudenay. Security of distance-bounding: A survey. *ACM Computing Surveys*, 2018.
[2] G. Avoine and C. H. Kim. Mutual distance bounding protocols. *IEEE Trans. Mob. Comput.*, 12(5):830–839, 2013.
[3] G. Bleumer. *Selective Forgery.* Springer US, Boston, MA, 2011.
[4] I. Boureanu, D. Gerault, and P. Lafourcade. Boxdb: Realistic adversary model for distance bounding. Cryptology ePrint Archive, Report 2018/1243, 2018. https://eprint.iacr.org/2018/1243.
[5] I. Boureanu and S. Vaudenay. Optimal proximity proofs. In *International Conference on Information Security and Cryptology*, pages 170–190. Springer, 2014.
[6] S. Brands and D. Chaum. Distance-bounding protocols. In *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, EUROCRYPT '93, pages 344–359, Berlin, Heidelberg, 1994. Springer-Verlag.
[7] T. Chothia, I. Boureanu, and L. Chen. Making contactless emv robust against rogue readers colluding with relay attackers. In *23rd International Conference on Financial Cryptography and Data Security (FC 19)*. International Financial Cryptography Association, February 2019.
[8] T. Chothia, F. D. Garcia, J. de Ruiter, J. van den Breekel, and M. Thompson. Relay cost bounding for contactless EMV payments. In R. Böhme and T. Okamoto, editors, *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, volume 8975 of *Lecture Notes in Computer Science*, pages 189–206, Puerto Rico, January 2015. Springer.
[9] Y. Desmedt, C. Goutier, and S. Bengio. Special uses and abuses of the fiat-shamir passport protocol. In *Advances in Cryptology - CRYPTO '87, A Conference on the*

Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings, pages 21–39, 1987.
[10] A. Dhar, I. Puddu, K. Kostiainen, and S. Capkun. ProximiTEE: Hardened SGX Attestation and Trusted Path through Proximity Verification. *IACR Cryptology ePrint Archive*, 2018:902, 2018.
[11] U. Dürholz, M. Fischlin, M. Kasper, and C. Onete. A formal approach to distance bounding RFID protocols. In *Information Security Conference ISC 2011*, volume 7001 of *Lecture Notes in Computer Science*, pages 47–62. Springer, 2011.
[12] EMVCo. Book C-2 kernel 2 specification v2.7. EMV contactless specifications for payment system, Feb, 2018.
[13] M. Fischlin and C. Onete. Terrorism in distance bounding: Modeling terrorist-fraud resistance. In *Applied Cryptography and Network Security*, ACNS'13, pages 414–431, Berlin, Heidelberg, 2013. Springer.
[14] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical and provably secure distance-bounding. In Y. Desmedt, editor, *Information Security*, pages 248–258, Cham, 2015. Springer.
[15] H. Kilinç and S. Vaudenay. Formal Analysis of Distance Bounding with Secure Hardware. In *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, pages 579–597, 2018.
[16] H. Kilinç and S. Vaudenay. Secure contactless payment. In *Information Security and Privacy - 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings*, pages 579–597, 2018.

A PROOFS

A.1 PayBCR – Security w.r.t. Strong Relaying

Theorem 4.5: *Consider the* $(\ell, z, n, q'_C, q'_R, q'_T, reader{-}coupled)$-v-DB *experiment in the strong relaying game for* PayBCR. *If* σ_1, σ_2 *are signatures unforgeable w.r.t. selective unforgeability (i.e., SUF-unforgeable) and AC is produced with a MAC that resists existential forgeries, then* PayBCR *is secure with respect to strong relaying. Concretely, the advantage of the adversary is:* $\frac{q_R^2}{2^{|N_R|}} + \frac{q_T^2}{2^{|\sigma'_1|}} + \frac{2{q_C}^2}{2^{|N_C|}}$, *where* $q_C = q'_C \cdot l$, $q_R = q'_R \cdot z$, $q_T = q'_T \cdot n$ *and* $|\cdot|$ *denotes the bit-length of a protocol message.*

Proof. This is a game-based proof. By $Pr[G]$, we mean the probability that $\mathcal{A}$ wins in the game G.

G_0: This game is the initial game, that is the strong relaying game against PayBCR. Unlike in this game, we do not use $\overline{C}$ and $\overline{R}$ for the target card and reader; instead, we simply use C and R.

G_1: This game is G_0, where no N_R value is indeed used more than once by any reader.

Let q_R be the number of N_R values issued by readers during the experiment (this is equal to the number of reader sessions). The probability that one N_R repeats is upper bounded by $\frac{{q_R}^2}{2^{|N_R|}}$, where $|N_R|$ is the bitlength of N_R. G_0 and G_1 are identical except for the failure event that two identical N_R values are used. So, we have $Pr[G_1] - Pr[G_0] \leq \frac{q_R^2}{2^{|N_R|}}$, which is negligible in the security parameter s if $|N_R|$ is in $\omega(s)$.

G_2: This game is the game G_1, with the difference that if the communication between parties R and T is too quick, when N_R is sent to T. Then, Ch aborts the experiment. That is, if $t(T \xleftarrow{N_R} R) < \Delta$ then Ch aborts (see Subsection 4.2.2), where Δ is the communication time between R and T. If the games continue, then $Pr[G_2] = Pr[G_1]$.

G_3: This game is the game G_2, with the difference that if the communication between parties R and T is too quick, when σ_1 is sent to R. Then, Ch aborts the experiment. That is, if $t(T \xrightarrow{t_1,\, \sigma_1} R) < \Delta$ then Ch aborts (see Subsection 4.2.2). If the games continue, then $Pr[G_3] = Pr[G_2]$.

G_4: This game is the game G_3 where σ'_1 does not repeat itself. Let q_T be the number of σ'_1 values issued by R during the experiment as this is equal to the number TPM sessions. The probability that one σ'_1 repeats is upper bounded by $\frac{{q_T}^2}{2^{|\sigma'_1|}}$, G_3 and G_4 are identical except for the failure event that two identical σ'_1 values are used, so we have $Pr[G_4] - Pr[G_3] \leq \frac{q_T^2}{2^{|\sigma'_1|}}$, which is negligible is negligible in the security parameter s – if $|\sigma'_1|$ is in $\omega(s)$.

G_5: This game is the game G_4, with the difference that if the communication between parties R and C is too quick when σ'_1 is sent. Then, Ch aborts the experiment. That is, if $t(R \xrightarrow{\sigma'_1} C) < t^{\frac{B}{2}}$ then Ch aborts (also, as point (7) in the adversary). If the games continue, then $Pr[G_5] = Pr[G_4]$.

G_6: This game is the game G_5, where no N_C value is used more than once by any card. Let q_C be the number of N_C values issued during the experiment, which is equal to the number of card sessions. The probability that one N_C repeats is upper bounded by $\frac{{q_C}^2}{2^{|N_C|}}$. So, G_5 and G_6 are identical except for the failure event that two identical N_C values are used, so we have $Pr[G_6] - Pr[G_5] \leq \frac{{q_C}^2}{2^{|N_C|}}$, which is negligible.

G_7: This game is the game G_6, with the difference that if the communication between parties R and C is too quick when N_C is sent. Then, Ch aborts the experiment. That is, if $t(R \xleftarrow{N_C,\, t_d} C) < t^{\frac{B}{2}}$ then Ch aborts (also, as point (7) in the adversary). If the games continue, then $Pr[G_7] = Pr[G_6]$.

G_8: This is the game G_7, except that the card never sends a value N_C that has previously been sent by an adversary through the *send* oracle. The idea behind this game transition is to eliminate the event E_G where A can randomly guess a value N_C in advance. Note that the number of calls to the send oracle with respect to N_C is bounded by the number of card sessions, q_C. Let q_C be the number of calls to the *send* oracle. This gives us that $Pr[E_G] \leq \frac{{q_C}^2}{2^{|N_C|}}$. Therefore, $Pr[G_6] - Pr[G_5] \leq \frac{{q_C}^2}{2^{|N_C|}}$, which is negligible.

G_9: This is the final game and is equivalent to the game G_8, with the difference that if the communication between parties R and T is too quick when N_C is sent. Then Ch aborts the experiment. That is, if $t(T \xleftarrow{N_C} R) < \Delta$ then Ch aborts (see Subsection 4.2.2). If the game continues, then $Pr[G_9] = Pr[G_8]$.

Now we look at the success probability of $\mathcal{A}$ in G_9, in which we assume the $Out_{[PC,T]} = 1$. Since $\mathcal{A}$ controls PC (onboard R), the output $\tau_{[PC,T]}$ can be tampered with by $\mathcal{A}$.

Note that $[W, X]$ (i.e., the bank) will check σ_1, σ_2 against t_1, t_2 and against other inputs that went into σ_1, σ_2 (e.g., N_C which is part of AC). These would all be provided – in the game– via $\tau_{[PC,T]}$ to $[W, X]$.

For $[W, X]$ (i.e., the bank) to output 1, the adversary needs to produce (t'_1, σ^*_1) and (t'_2, σ^*_2) that put inside the forged $\tau_{[PC,T]}$ and with σ^*_i being valid signatures by T on t'_i (with $i \in \{1, 2\}$).

Since $\mathcal{A}$ controls R, note that $\mathcal{A}$ can choose N_R (one that has not produced been input to another σ_1, given the game in which we are). Also, even if he does not control C, $\mathcal{A}$ can also choose one such fresh N_C – since it controls the channel between C and R. If he chooses one such fresh N_C, it also needs to forge the MAC AC

(which will be part of $\tau_{[PC,T]}$). If the AC is produced with a MAC that resists existential forgery, then this is negligible.

Further, as per the model, $\mathcal{A}$ does not control T, so $\mathcal{A}$ cannot choose t_1, t_2 or the randomness of T. The former are timed-structures that are in part fixed prior to the attack starting (e.g., they contain global-clock value). As such, if σ_1 and σ_2 are unforgeable w.r.t. selective unforgeability (i.e., SUF-unforgeable), then (t_1', σ_1') and (t_2', σ_2') as per the above can only be produced with negligible probability.

□

A.2 PayCCR – Security w.r.t. Strong Relaying

Theorem 4.6: *Consider the* $(\ell, z, n, q_C', q_R', q_T', reader{-}coupled)$-v-DB *experiment in the strong relaying game for* PayCCR. *If* σ_1, σ_2 *are signatures unforgeable w.r.t. selective unforgeability (i.e., SUF-unforgeable), then* PayCCR *is secure with rwith respect to strong relaying. Concretely, the advantage of the adversary is:* $\frac{q_R^2}{2^{|N_R|}} + \frac{q_T^2}{2^{|\sigma_1|}} + \frac{2{q_C}^2}{2^{|N_C|}}$, *where* $q_C = q_C' \cdot l$, $q_R = q_R' \cdot z$ *and* $q_T = q_T' \cdot n$ *and* $|\cdot|$ *denotes the bit-length of a protocol message.*

Proof. This is a game-based proof. By $Pr[G]$, we mean the probability that $\mathcal{A}$ wins in a game G.

G_0: This game is the initial game, that is the strong relaying game against PayCCR. Unlike in this game, we do not use $\overline{C}$ and $\overline{R}$ for the target card and reader; instead, we simply use C and R.

G_1: This game is G_0, where no where no N_R value is indeed used more than once by any reader.

Let q_R be the number of N_R values issued by readers during the experiment (this is equal to the number of reader sessions). The probability that one N_R repeats is upper bounded by $\frac{q_R^2}{2^{|N_R|}}$. So, G_0 and G_1 are identical except for the failure event that two identical N_R values are used, so we have $Pr[G_1] - Pr[G_0] \leq \frac{q_R^2}{2^{|N_R|}}$, which is negligible in the security parameter s – if $|N_R|$ is in $\omega(s)$.

G_2: This game is the game G_1, with the difference that if the communication between parties R and T is too quick, when N_R is sent to T. Then, Ch aborts the experiment. That is, if $t(T \xleftarrow{N_R} R) < \Delta$ then Ch aborts (see Subsection 4.2.2), where Δ is the communication time between R and T. If the games continue, then $Pr[G_2] = Pr[G_1]$.

G_3: This game is the game G_2, with the difference that if the communication between parties R and T is too quick, when σ_1 is sent to R. Then, Ch aborts the experiment. That is, if $t(T \xrightarrow{t_1,\ \sigma_1} R) < \Delta$ then Ch aborts (see Subsection 4.2.2), where Δ is the communication time between R and T. If the games continue, then $Pr[G_3] = Pr[G_2]$.

G_4: This game is the game G_3 where σ_1 does not repeat itself. Let q_T be the number of σ_1 values issued by R during the experiment as this is equal to the number TPM sessions. The probability that one σ_1 repeats is upper bounded by $\frac{{q_T}^2}{2^{|\sigma_1|}}$. G_3 and G_4 are identical except for the failure event that two identical σ_1 values are used, so we have $Pr[G_4] - Pr[G_3] \leq \frac{q_{\sigma_1}^2}{2^{|\sigma_1|}}$, which is negligible in the security parameter s – if $|\sigma_1|$ is in $\omega(s)$.

G_5: This game is the game G_4, with the difference that if the communication between parties R and C is too quick when σ_1 is sent to C. Then, Ch aborts the experiment. That is, if $t(R \xrightarrow{\sigma_1'} C) < t^{\frac{B}{2}}$ then Ch aborts (also, as point (7) in the adversary). If the games continue, then $Pr[G_5] = Pr[G_4]$.

G_6: This game is the game G_5, where no N_C value is used more than once by any card. Let q_C be the number of N_C values issued by readers during the experiment encapsulating this game as this is equal to the number of card sessions. The probability that one N_C repeats is upper bounded by $\frac{{q_C}^2}{2^{|N_C|}}$. G_5 and G_6 are identical except for the failure event that two identical N_C values are used, so we have $Pr[G_6] - Pr[G_5] \leq \frac{{q_C}^2}{2^{|N_C|}}$, which is negligible in the security parameter s – if $|N_C|$ is in $\omega(s)$.

G_7: This game is the game G_6, with the difference that if the communication between parties R and C is too quick, then Ch aborts the experiment when N_C is sent to R. Then, Ch aborts the experiment. That is, if $t(R \xrightarrow{\sigma_1'} C) < t^{\frac{B}{2}}$ then Ch aborts (also, as point (7) in the adversary). If the games continue, then $Pr[G_7] = Pr[G_6]$.

G_8: This is the game G_7, except that the card never sends a value N_C that has previously been sent by an adversary through the *send* oracle. The idea behind this game transition is to eliminate the event E_G where A can randomly guess a value N_C in advance. Note that the number of calls to the send oracle with respect to N_C is bounded by the number of card sessions, q_C. Let q_C be the number of calls to the send oracle. This gives us that $Pr[E_G] \leq \frac{{q_C}^2}{2^{|N_C|}}$. Therefore, $Pr[G_6] - Pr[G_5] \leq \frac{{q_C}^2}{2^{|N_C|}}$, which is negligible in the security parameter s – if $|N_C|$ is in $\omega(s)$.

G_9: This is the final game and is equivalent to the game G_8, with the difference that if the communication between parties R and T is too quick, when N_C is sent to T. Then, Ch aborts the experiment. That is, if $t(T \xleftarrow{N_C} R) < \Delta$ then Ch aborts (see Subsection 4.2.2), where Δ is the communication time between R and T. If the games continue, then $Pr[G_9] = Pr[G_0]$.

Now we look at the success probability of $\mathcal{A}$ in G_9, in which we assume the $Out_{[PC,T]} = 1$ (since PC on board R is controlled by the attacker). I.e., since $\mathcal{A}$ controls PC, the private output $\tau_{[PC,T]}$ can be tampered with by $\mathcal{A}$.

Note that $[W, C]$ (i.e., the card) will check σ_1, σ_2 against t_1, t_2 and against other inputs that went into σ_1, σ_2 such as N_C. Formally, in the game, $[W, C]$ get the private output $\tau_{[PC,T]}$

For $[W, C]$ (i.e., the card) to output 1, the adversary needs to produce (t_1', σ_1^*) and (t_2', σ_2^*) to put in the forged $\tau_{[PC,T]}$ and with σ_i^* being valid signatures by T on t_i' (with $i \in \{1, 2\}$).

Since $\mathcal{A}$ controls R, note that $\mathcal{A}$ can choose N_R (one that has not produced been input to another σ_1, given the game in which we are). Also, $\mathcal{A}$ cannot choose N_C – since this is produced by C within $[W, C]$.

But, as per the model, $\mathcal{A}$ does not control T, so $\mathcal{A}$ cannot choose t_1, t_2 or the randomness of T. The former are timed-structures that are in part fixed prior to the attack starting (e.g., they contain global-clock value). As such, if σ_1 and σ_2 are unforgeable w.r.t. selective unforgeability (i.e., SUF-unforgeable), then (t_1', σ_1') and (t_2', σ_2') as per the above can only be produced with negligible probability.

□